Homework Form for Data Mining *

Alex Clemmer

February 1, 2012

Overview

This is a sample latex file to use for completing assignments. This particular file is not required. In fact, there are many cool ways to spruce up this plain look. Feel free to use them.

Q1: Birthday "Paradox"

A: For domain $n = 1000$, it took 58 random trials.

B: Please see figure 1.

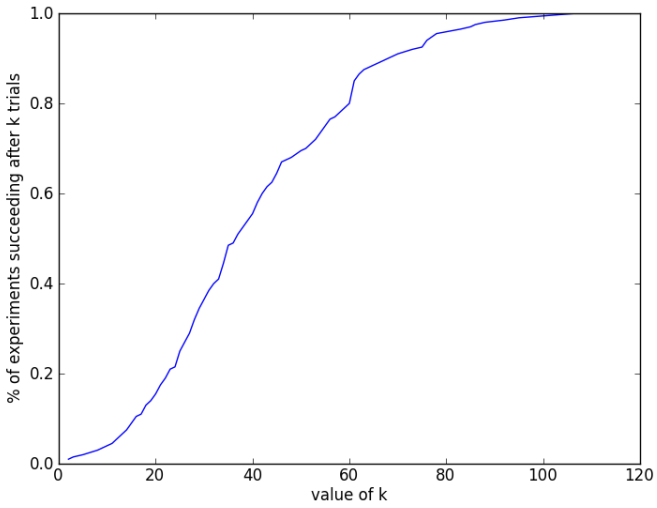

Figure 1: The % of experiments requiring k tests before collision, plotted as a function of k.

C: For some $m = 200$ random variables $X_1 \ldots X_m$ representing the outcomes of m random repetitions of the experiment, the expected value $\mathbf{E}[\vec{X}] = 38.005$

*CS 6955 Data Mining; Spring 2012 Instructor: Jeff M. Phillips, University of Utah

D: I check uniqueness using a bit vector of length n, where each place is 0 if we haven't seen the corresponding element before, and is 1 if we have. We use a Mersenne Twister as the random number generator. We run the experiment, continuously checking if the generated number corresponds to an already-filled bit in the bit vector.

As long as this bit vector fits in memory, it should scale pretty well. Experimentally this bears out well: in figure 2, we show experimentally that, for $n = 1,000,000$, even when we increase m by orders of 10 starting at $m = 10$, the algorithm still runs in roughly linear time.

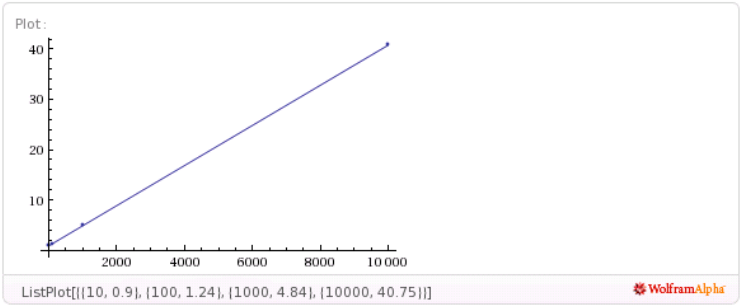

Figure 2: The running time (x-axis) increases by successive powers of 10; the y-axis denotes the running time of seconds it took to complete.

Q2: Coupon Collectors

A: For the domain $n = 60$, the required trials $k = 198$.

B: As we can see in figure 3, the highest bar was pinged 10 times.

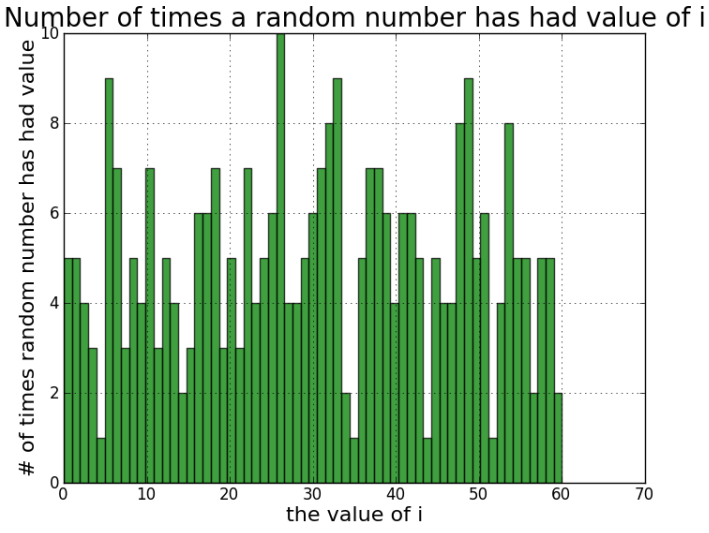

Figure 3: The number of times a random number ends up with value i.

C: Please see figure 4.

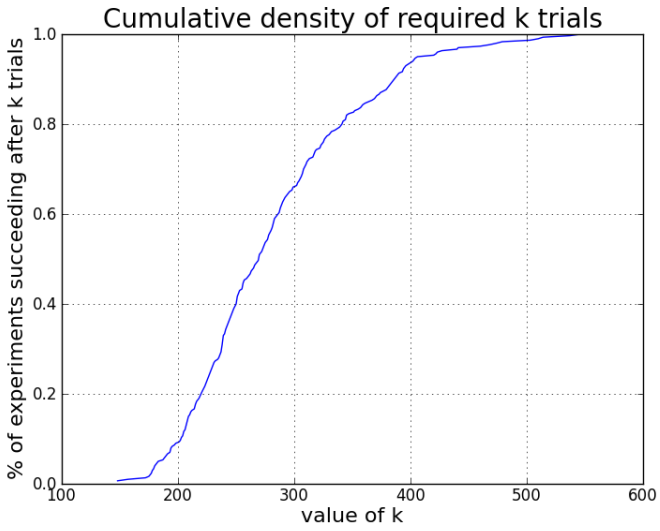

Figure 4: The number of times a random number ends up with value i.

D: For some $m = 300$ random variables $X_1 \ldots X_m$ representing the outcomes of m random repetitions of the experiment, the expected value $\mathbf{E}[\vec{X}] = 283.61$

E: Our "basic" algorithm again used an n-length bit vector to check for collisions, but this time we also kept track of the number of unique elements; when this number reaches n, we halt the algorithm. We again used a Mersenne Twister for random number generation.

This is not going to scale as well. In figure 5, we increase n by orders of 10, for constant $m = 300$. Note that it is larger than linear. The main reason is that the expected number of times we need to generate a number before we have generated every number in a domain of n is going to be asymptotically much larger than linear with respect to n.

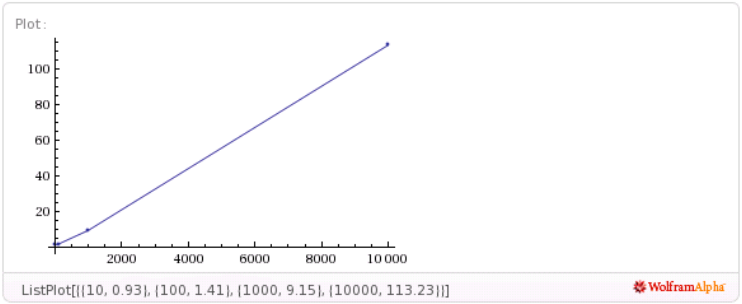

Figure 5: The running time (x-axis) increases by successive powers of 10; the y-axis denotes the running time of seconds it took to complete.

How much this can be sped up depends on what the real bottleneck is. If the real bottleneck is generating the random numbers sequentially (and I suspect it is), then you can probably improve performance by parallelizing the random number generation. You might begin, for example, by firing off some t different "mapper" threads, each of which is constantly generating numbers from Mersenne Twister, simply dumping them to a common place (*e.g.*, standard out). You could then

employ a single "reducer" thread, whose only job is to simply check whether we've seen a particular number against the bit vector.

Notably, this does not completely eliminate the runtime issues. What it *does* do is shift the bottleneck to communication. It's not a solution to the problem; you're really just switching one bad problem for another.

Q3: Analysis

A: The explicit formula for $n = 1000$ is

$$1 - (1 - \frac{1}{1000})^{\binom{k}{2}} = 1/2$$

Assuming k is real, we can reduce to:

$$1 - ((\frac{3}{10})^{k^2-k})^{3/2}\sqrt{37^{k^2-k}} = 1/2$$

This is polynomial equation which we can solve the usual way, giving us:

$$k = 37.727$$

B: The explicit formula for $n = 60$ is a straightforward calculation. I actually got my formula from Wikipedia, since the formulation in the notes gave me a weird result. Their formulation is also convincing (I think that the series analysis in the notes is wrong). Anyway, it gives us a much closer result to empirical truth:

$$\begin{aligned} k &= n \log n + \gamma n + \frac{1}{2} + C_1 \\ &= (1 + \gamma) 60 \log 60 \\ &\approx 280.7806 \end{aligned}$$