CS 6140: Data Mining Assignment 1

Hash Functions and PAC Algorithms

Yash Gangrade (u1143811), MS in Computing 16th January 2019

Contents

1 Birthday Paradox

Ans A) It took around 53 random trials for me to get the two that have the same value in the domain of size 5000. We are using uniform distribution over [n].

Ans B) The plot for the Cumulative Density function plot when m = 300 is shown below:

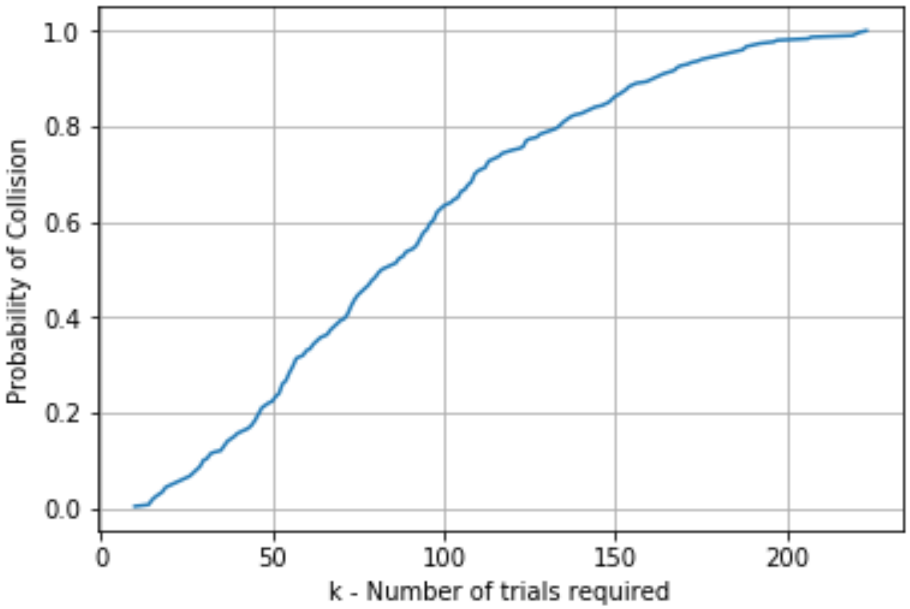

Figure 1: CDF Plot for m = 300

Ans C) The empirical expected estimate of k is 86.3.

Ans D) I implemented this entire experiment in Python (version 3.6). For m = 300 trials, it took around 184 ms. A brief summary of implementation is discussed below.
Library named random in python was used to generate the random numbers and matplotlib was used to generate the required plots. For the first part, simply a set was used to store all the generated random numbers. Every time a new random number is generated, a check is performed for the collision. If there is no collision, this number is added into the set and operation is continued, otherwise if there is a collision then we simply return the number of trials it took this point. For part b, we call the part a function for 300 times and then sort the resulting array after getting all the distinct elements from it. It is then plotted using the plotting functions in matplotlib. For part c, we are simply adding up all values of k and then dividing it by m. For part d, we have to run the above experiments multiple times for different m and n values. It's a time-intensive process.

The plot of run time is shown on the next page.

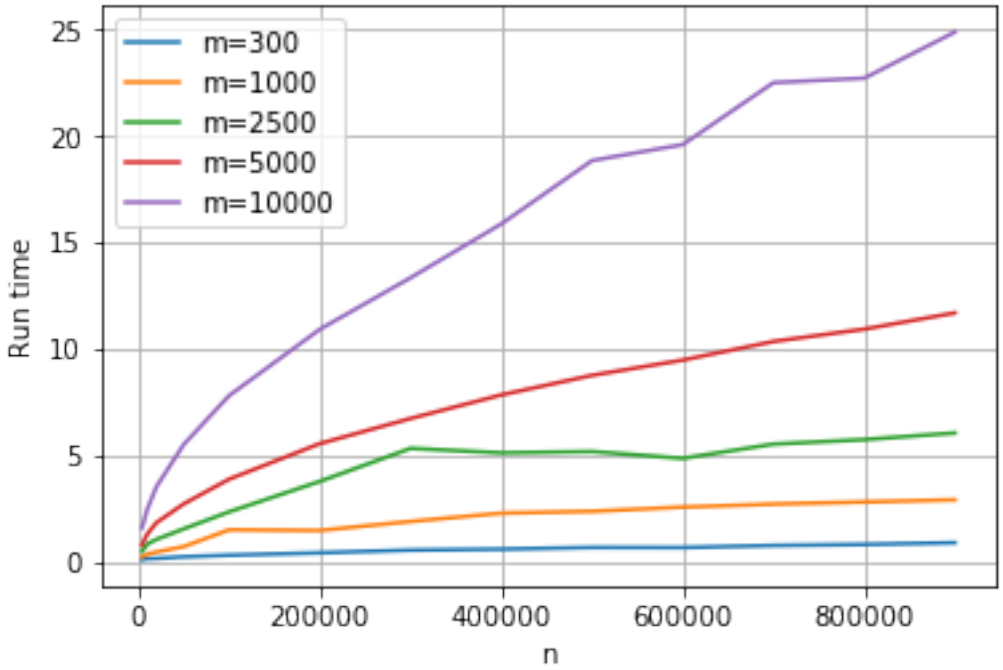

Figure 2: Run time as a function of n for different values of m

2 Coupon Collectors

Ans A) It took around 1890 trials to generate random numbers in the domain $[n]$ until every value $i \in [n]$ has at least one random number equal to i.

Ans B) The plot for the Cumulative Density function plot when m = 400 is shown below:

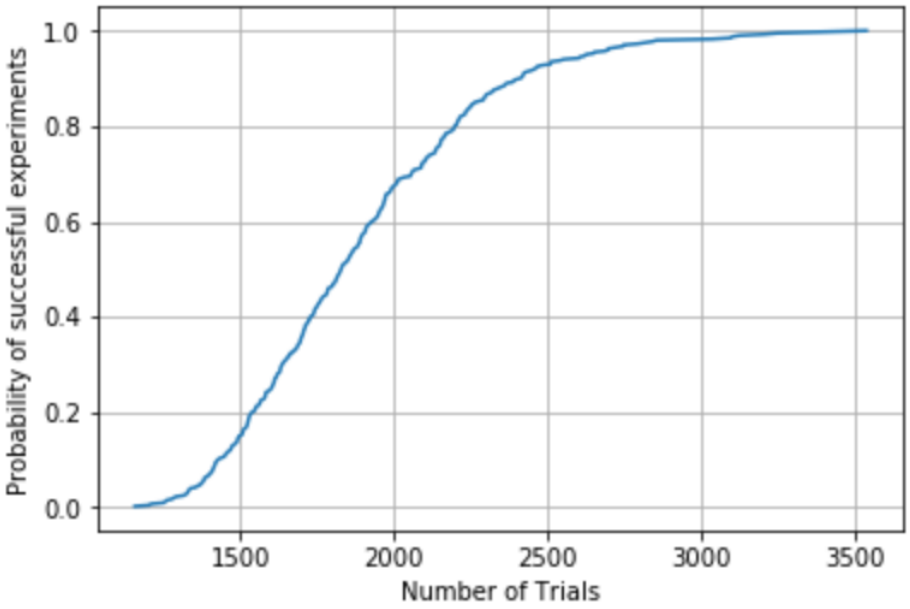

Figure 3: CDF Plot for m = 400

Ans C) The empirical expected value of k comes out to be 1888.

Ans D) I implemented this entire experiment in Python (version 3.6). For n = 300 and m = 400 trials, it took around 1.7 seconds. A brief summary of the implementation is discussed below.
Library named random in python was used to generate the random numbers and matplotlib was used to generate the required plots. For the first part, simply a set was used to store all the generated random numbers. Every time a new random number is generated, a check is performed on the size of the set. If it is equal to n then we return the number of generating trials it took till this point. Otherwise, this number is added into the set and operation is continued. For part b, we call the part a function for 400 times and then sort the resulting array after getting all the distinct elements from it. It is then plotted using the plotting functions in matplotlib. For part c, we are simply adding up all values of k and then dividing it by m. For part d, we have to run the above experiments multiple times for different m and n values. It's a time-intensive process. The plot of run time is shown on the next page.

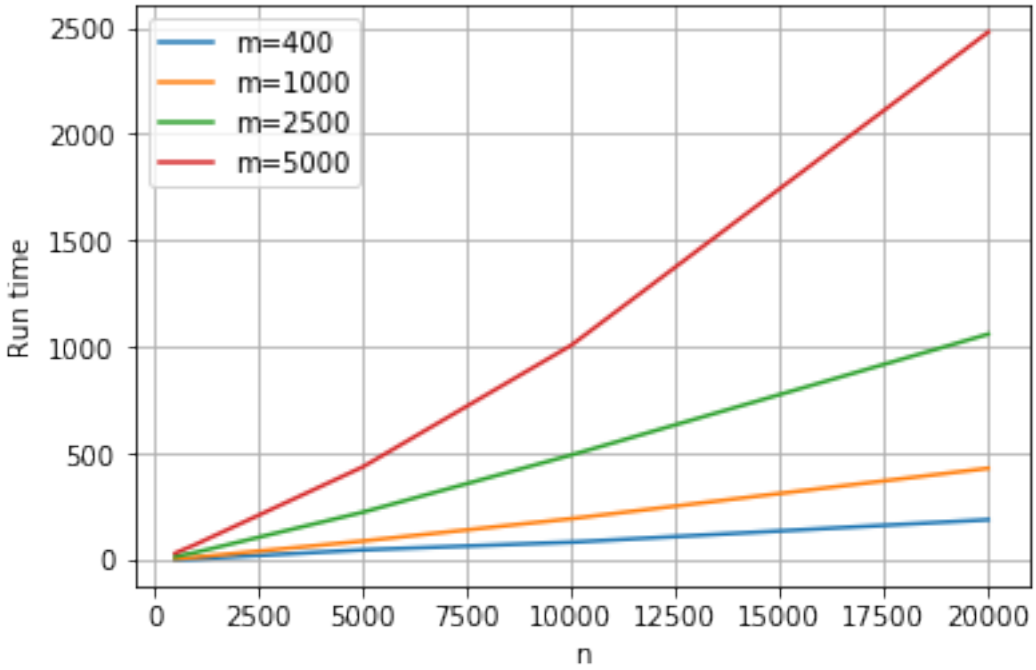

Figure 4: Run time as a function of n for different values of m

3 Comparing Experiments to Analysis

Ans A) There are multiple formulas available to use for this type of questions. I am using the formula from the slides. The equation 1 refers to the probability of no collision.

$$\mathrm{P}(NoCollision) = \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{n-3}{n} \cdots \tag{1}$$

$$\implies \mathrm{P}(NoCollision) = \prod_{i=1}^{k-1} \left(\frac{n-i}{n}\right) \tag{2}$$

$$\implies \mathrm{P}(Collision) = 1 - \prod_{i=1}^{k-1} \left(\frac{n-i}{n}\right) \tag{3}$$

In our question, we have $n = 5000$ and it is also given that the probability of collision should be at least 0.5. So putting it all together, we get,

$$1 - \prod_{i=1}^{k-1} \left(1 - \frac{i}{5000}\right) = 0.5$$

$$\prod_{i=1}^{k-1} \left(1 - \frac{i}{5000}\right) = 0.5$$

Now, we have to try different k values to get the best estimate. For this I wrote a small program in python to compute probability for different k. So if the probability of collision needs to be at least greater than 0.5, k comes out to be 85. In the experiments stated in part 1, we are getting empirical k values around 86.3 which is very close to the theoretical analysis.

Ans B) From the formulas discussed in the lecture notes, we have the expected number of trials to witness i^{th} coupon if we already have $i-1$ coupons is

$$t_i = \frac{n}{n-i+1}$$

Therefore, we can count the total number of trials (T) before all the elements are witnessed as follows:

$$k = T = \sum_{i=1}^{n} \frac{n}{n-i+1} = n \cdot \sum_{i=1}^{n} \frac{1}{i} \approx n(\gamma + \ln n)$$

Now, putting the values $n = 300$ and $\gamma = 0.577$ in the above formula, we get

$$\begin{aligned} k &= n(\gamma + \ln n) \\ k &= 300(0.577 + \ln 300) \\ k &\approx 1884.235 \end{aligned}$$

Through our experiments, we got the empirical estimate of k around 1888 which is very close to the results from formula.

4 BONUS: PAC Bounds

Ans For the first part of the question, we are given,

$$P_r\left[\left|\mu - \frac{1}{n}\right| \geq 0.04\right] \leq \delta$$

In the lecture, we have seen that the Chernoff bounds is as follows

$$P_r\left[|A - E[A]| > \epsilon\right] \leq 2 \cdot exp\left(\frac{-2k\epsilon^2}{\Delta^2}\right)$$

For the first part, we can compare the two equations and put the values i.e. $\Delta = n$, $\epsilon = 0.04$ into the equation. Then we compare the RHS of both equations and we get,

$$\begin{aligned} 2 \cdot exp\left(\frac{-2k\epsilon^2}{\Delta}\right) &= \delta \\ 2 \cdot exp\left(\frac{-2k(0.04)^2}{n^2}\right) &= \delta \\ -\frac{0.0016k}{n^2} &= \ln\frac{\delta}{2} \end{aligned}$$

$$\implies \boxed{k = 625n^2 \ln\left(\frac{\delta}{2}\right)}$$

Secondly, we want to see the new value of k if the error rate changes by a factor of 10 i.e. $\epsilon = 0.004$, we have,

$$\begin{aligned} 2 \cdot exp\left(\frac{-2k\epsilon^2}{\Delta}\right) &= \delta \\ 2 \cdot exp\left(\frac{-2k(0.004)^2}{n^2}\right) &= \delta \\ -\frac{0.000016k}{n^2} &= \ln\frac{\delta}{2} \end{aligned}$$

$$\implies \boxed{k = 62500n^2 \ln\left(\frac{\delta}{2}\right)}$$

Essentially, k increases by a factor of 100 when the error percentage goes down by a factor of 10. Also, these bounds on k are light upper bounds.