Asmt 1: Hash Functions and PAC Algorithms

sushmitha.sn@utah.edu - u1265043

January 2020

1 Birthday Paradox (35 points)

Consider a domain of size $n = 5000$.

A: (5 points) Generate random numbers in the domain $[n]$ until two have the same value. How many random trials did this take? We will use k to represent this value. Solution:
It took me 87 random trials. k = 87

B: (10 points) Repeat the experiment $m = 300$ times, and record for each time how many random trials this took. Plot this data as a *cumulative density plot* where the x-axis records the number of trials required k, and the y-axis records the fraction of experiments that succeeded (a collision) after k trials. The plot should show a curve that starts at a y value of 0, and increases as k increases, and eventually reaches a y value of 1.
Solution:

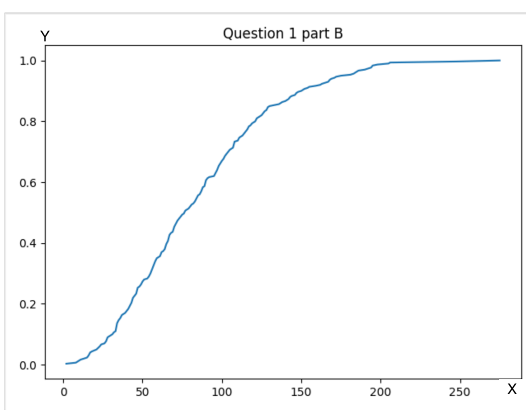

Figure 1: Plot of number of trials required k Vs fraction of experiments that succeeded after k trials

Above is the graph of number of trials required k Vs fraction of experiments that succeeded after k trials. X axis of the graph represents k trials. Y axis represents (cumulative) fraction of experiments succeeded after k trials.

C: (10 points) Empirically estimate the expected number of k random trials in order to have a collision. That is, add up all values k, and divide by m.
Solution:
Expected number of k random trials = $\frac{\sum k}{m}$ m = 300
Empirical value 84.82333333333334

D: (10 points) Describe how you implemented this experiment and how long it took for $m = 300$ trials.
Time taken for 300 trials 0.055513858795166016
I have a list which stores generated random numbers. While the generated random number is not in the list, I keep calling the random number generator function inside the while loop and i increment the variable k every time I call randint() function. Once I come across a number which already exists in the list, there is a collision. Now the while condition is false because the number is already there in the list, hence it comes out of while loop. I store the value k as a key in dictionary with value as number of times collision occurred after k trials. In other words, if an entry for k already exists in the dictionary, I simply increment its value by 1. Otherwise, I set the value to 1. I repeat the above process in a for loop with range 0 to 300.
code:

```
lst_300_trials_x = []
start_time = time.time()
for i in range  (0,300):
    num = randint(1, 5000)
    k = 1
    set1 = []
    while num not in set1:
        set1.append(num)
        num = randint(1, 5000)
        k += 1
    lst_300_trials_x.append(k)
dict_y1 = {}
for item in lst_300_trials_x:
    dict_y1[item] = lst_300_trials_x.count(item)

#convert dictionary entry into x and y axis
x = []
y = []
```

```
for key, value in sorted(dict_y1.items()):
    x.append(key)
    if len(y)>0:
        prev = y[-1]
        y.append(prev+value/300)
    else:
        y.append(value/300)
end_time = time.time()
plt.plot(x,y)
plt.title("Question 1 part B")
plt.show()
```

Show a plot of the run time as you gradually increase the parameters n and m. (For at least 3 fixed values of m between 300 and 10,000, plot the time as a function of n.) You should be able to reach values of n = 1,000,000 and m = 10,000. Solution:

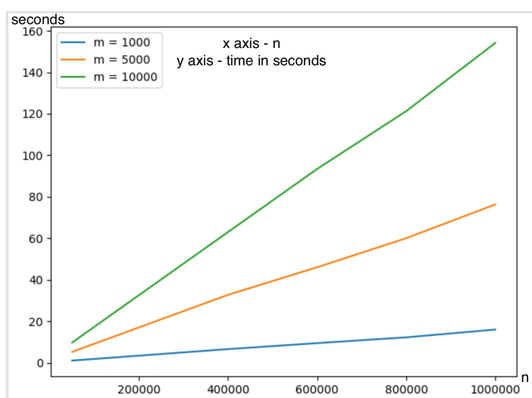

Figure 2: Plot of n Vs t

2 Coupon Collectors (35 points)

Consider a domain $[n]$ of size $n = 300$.

A: (5 points) Generate random numbers in the domain $[n]$ until every value $i \in [n]$ has had one random number equal to i. How many random trials did this take? We will use k to represent this value. Solution:
Number of random trials to get all the numbers in the domain [n] 2078.
k = 2078

B: (10 points) Repeat step **A** for $m = 400$ times, and for each repetition record the value k of how many random trials we required to collect all values $i \in [n]$. Make a cumulative density plot as in **1.B**.

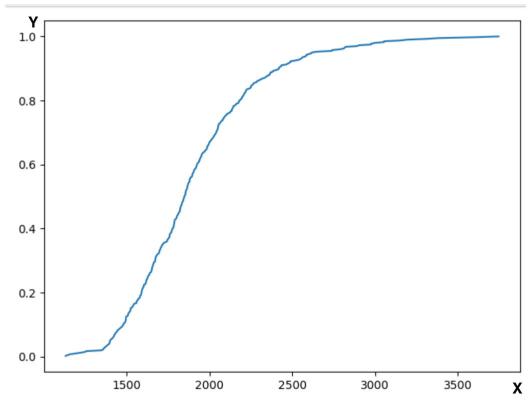

Figure 3: Plot of number of trials required k to get all the values in n(x-axis) Vs fraction of experiments that succeeded after k trials(y-axis)

C: (10 points) Use the above results to calculate the empirical expected value of k.
Solution:
expected value of k $= \frac{\sum k}{m}$ m = 400
Empirical value 1911.3125

D: (10 points) Describe how you implemented this experiment and how long it took for $n = 300$ and $m = 400$ trials.
Show a plot of the run time as you gradually increase the parameters n and m. (For at least 3 fixed values of m between 400 and 5,000, plot the time as a function of n.) You should be able to reach $n = 20{,}000$ and $m = 5{,}000$. Solution:
I ran the above code for different values of m and n. Below is the plot I have got, It took me 0.99462 seconds to run 400 trials.
Implementation: I have a set to store generated random numbers in the domain [n]. Initially it will be empty. While the set size is not equal to n, I keep generating the random number and i increment variable k. Once I have generated all the number in the domain[n], length of set will be equal to n. Hence while loop condition fails and it comes out of while loop. I store the value of variable k as key in the dictionary and value as number of times we have got k. We do the above process for m times. After m trials, we can plot the graph using dictionary key as x axis and its corresponding value as y axis.

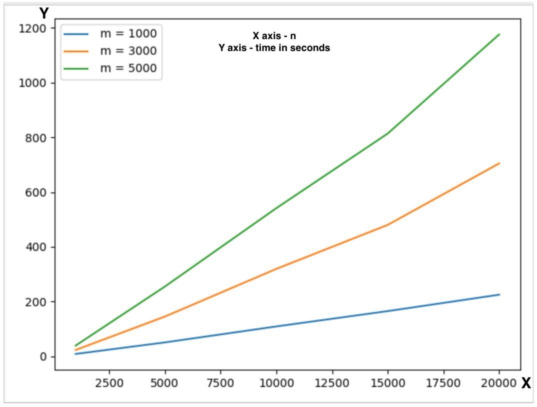

Figure 4: Plot of time in seconds(y-axis) n Vs n(x-axis)

```
Implementation
dict_xy = {}
lst_400_trials_k = []
start_time = time.time()
for i in range(0,400):
    set1 = set([])
    num = randint(1,300)
    k = 1

    while(len(set1)) != 300:
        if num not in set1:
            set1.add(num)
        num = randint(1,300)
        k+=1
    lst_400_trials_k.append(k)
    if k not in dict_xy:
        dict_xy[k] = 1
    else:
        dict_xy[k] += 1

#convert dict_xy{} into x axis and y axis
x = []
y = []
for key,value in sorted(dict_xy.items()):
    x.append(key)
    if len(y)>0:
        prev = y[-1]
        y.append(prev+value/400)
    else:
```

```
        y.append(value/400)
endtime = time.time()
print("Time taken to run 400 trials", endtime-start_time)
plt.plot(x,y)
plt.show()
```

3 Comparing Experiments to Analysis (30 points)

A: (15 points) Calculate analytically (using formulas from the notes in **L2** or M4D book) the number of random trials needed so there is a collision with probability at least 0.5 when the domain size is $n = 5000$. There are a few formulas stated with varying degree of accuracy, you may use any of these – the more accurate formula, the more sure you may be that your experimental part is verified, or is not (and thus you need to fix something).
[Show your work, including describing which formula you used.]

How does this compare to your results from **1.C**? Solution:
Given domain size = 5000
Probability of getting a number in the domain after one trial = 1
Probability of getting the same number in next trial = $\frac{1}{n}$
Probability of not getting the same number is = $1 - \frac{1}{n}$
There are ${}^kC_2$ such pairs. Probability of not having collision for all the ${}^kC_2$ pairs is $(1 - \frac{1}{n})^{{}^kC_2}$
Probability of having a collision for such pairs is $1 - (1 - \frac{1}{n})^{{}^kC_2}$ and this should be more than or equal to 0.5.

Using the formula:

$$1 - (1 - \frac{1}{n})^{{}^kC_2} \geq 0.5$$

Substitute n = 5000

$$1 - (1 - \frac{1}{5000})^{{}^kC_2} \geq 0.5$$

On simplifying

$$1 - (\frac{4999}{5000})^{{}^kC_2} \geq 0.5$$

$$1 - 0.5 \geq (\frac{4999}{5000})^{{}^kC_2}$$

$$0.5 \geq (\frac{4999}{5000})^{{}^kC_2}$$

$$(\frac{4999}{5000})^{{}^kC_2} \leq 0.5$$

Taking log on both the sides

$$log((\frac{4999}{5000})^{{}^kC_2}) \leq log(0.5)$$

$$^kC_2(log(\frac{4999}{5000})) \leq log(0.5)$$

$$\frac{k*(k-1)}{2*1}(log(0.9998)) \leq log(0.5)$$

log(0.9998) = -0.0002885678659, log(0.5) = -1

$$\frac{k*(k-1)}{2*1} * (-0.0002886) \leq -1$$

$$k*(k-1)*(-0.0002886) \leq -2$$

$$(k^2 - k)*(0.0002886) \geq 2$$

$$(k^2 - k) \geq \frac{2}{0.0002886}$$

$$(k^2 - k) \geq \frac{2}{0.0002886}$$

$$(k^2 - k) \geq 6930.006$$

On Solving the above quadratic equation,

$$k \leq -82.748165 \ or \ k \geq 83.748165$$

Since k is the number of random trials, it cannot be negative. Hence, $k \geq 83.74816$ and k is the number of trials hence it has to be integer. So the value of k is 84.

The value that I got in 1.C is 84.823 which is slightly bigger yet very close to the value of k (i,e 84) I got in this section.

B: (15 points) Calculate analytically (using formulas from the notes in **L2** or M4D book) the expected number of random trials before all elements are witnessed in a domain of size $n = 300$? Again, there are a few formulas you may use – the more accurate, the more confidence you might have in your experimental part.
[Show your work, including describing which formula you used.]

Solution:
There are 300 different numbers. On each trial we have equal probability of $\frac{1}{300}$ of getting each number. We want to collect all the numbers from 1 to 300. How many trials (k) we expect to have before collecting all the numbers?
Let r_i be the expected number of trials we need to take before receiving exactly i distinct numbers. Let $r_0 = 0$ and set $t_i = r_i - r_{i-1}$ to measure the expected number of trials between getting i - 1 distinct numbers and i distinct numbers.

Clearly $r_1 = t_1 = 1$ which means the first trial always yields a new number. Expected number of trials to get all the numbers is $T = \sum_{i=1}^{n} t_i$

To measure t_i we will define p_i as the probability that we get a new number after already having $i-1$ distinct numbers. Thus $t_i = 1/p_i$. And $p_i = \frac{(n-i+1)}{n}$. Now,

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

Using the formula:

$$T = n * \sum_{i=1}^{n} \frac{1}{i}$$

expected number of random trials before all elements are witnessed in a domain of size n=300 is

$$= 300 * \sum_{i=1}^{n} \frac{1}{i}$$

$$= 300 * \sum_{i=1}^{n} \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{300}$$

On Simplifying the above equation, we get: 1884.9

How does this compare to your results from **2.C**? The value I got in 2.C is 1911.3125 which is slightly bigger than the value 1884.9 yet it is very close.

4 BONUS : PAC Bounds (2 points)

Consider a domain size n and let k be the number of random trials run, where each trial obtains each value $i \in [n]$ with probability $1/n$. Let f_i denote the number of trials that have value i. Note that for each $i \in [n]$ we have $\mathrm{E}[f_i] = k/n$. Let $\mu = \max_{i \in [n]} f_i/k$.

Consider some parameter $\epsilon \in (0, 1)$. As a function of parameter ϵ, how large does k need to be for $\Pr[|\mu - 1/n| \geq \epsilon] \leq 0.05$? That is, how large does k need to be for *all* counts to be within $(\epsilon \cdot 100)\%$ of the average with probability 0.05? *(Fine print: you don't need to calculate this exactly, but describe a bound as a function of ϵ for the value k which satisfies PAC property. Chapter 2.3 in the M4D book should help.)*

How does this change if we want $\Pr[|\mu - 1/n| \geq \epsilon] \leq 0.005$ (that is, only 0.005 probability of exceeding ϵ error)?

[Make sure to show your work.]