Asmt 1: Hash Functions and PAC Algorithms

Turn in (**a pdf**) through Canvas by 2:45pm:
Wednesday, January 15

Overview

In this assignment you will experiment with random variation over discrete events.

It will be very helpful to use the analytical results and the experimental results to help verify the other is correct. If they do not align, you are probably doing something wrong (this is a very powerful and important thing to do whenever working with real data).

As usual, it is recommended that you use LaTeX for this assignment. If you do not, you may lose points if your assignment is difficult to read or hard to follow. Find a sample form in this directory: `http://www.cs.utah.edu/~jeffp/teaching/latex/`

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.

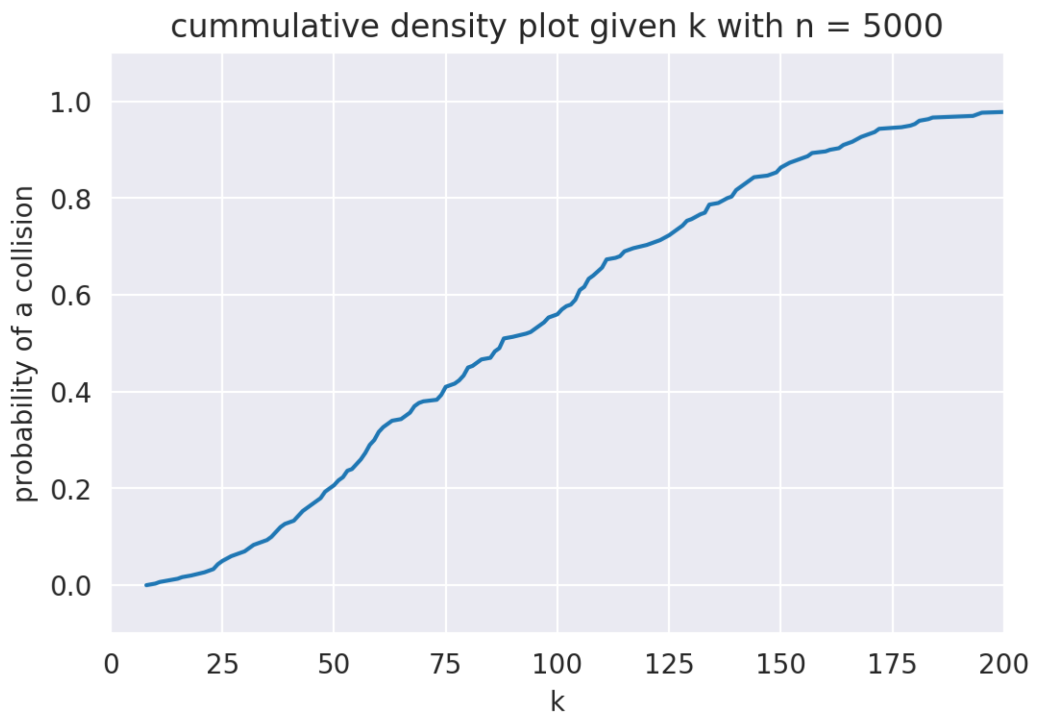
128 trials to get a repeat when the expected amount of trials to get a repeat is 100

```
import random
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
import numpy as np
import math
import time

def collision(size):
    random_numbers_set = set()
    while True:
        nextRan = random.randint(1,size)
        if nextRan not in random_numbers_set:
            random_numbers_set.add(nextRan)
            continue
        else: # collision occurs
            break
    return len(random_numbers_set)+1 # add one at collision step

n = 5000
print('%d trials to get a collision'%collision(n))
print('when the expected amount of trials is %d'%math.sqrt(2*n))
```

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.
93 trials to get a collision when the expected amount is 100

```
m = 300 #number of trials
n= 5000 #number of domain
p = []
k = []
prob = 1/m
for q in range(0, m):
    nextRand = collision(n)
    k.append(nextRand)
k = sorted(k)
for z in range(len(k)):
    p.append(k.index(k[z])*prob)
print('%d trials to get a collision'% np.mean(k))
print('when the expected amount is %d'%math.sqrt(2*n))

sns.set_style("darkgrid")
mpl.rcParams['figure.figsize'] = (6,4)
mpl.rcParams['figure.dpi'] = 200
plt.title("cummulative density plot given k with n = %d" %n)
plt.xlabel("k")
plt.ylabel("probability of a collision")
plt.xlim([0,200])
plt.ylim([-0.1,1.1])
plt.plot(np.array(k), np.array(p))
plt.show()
```



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.

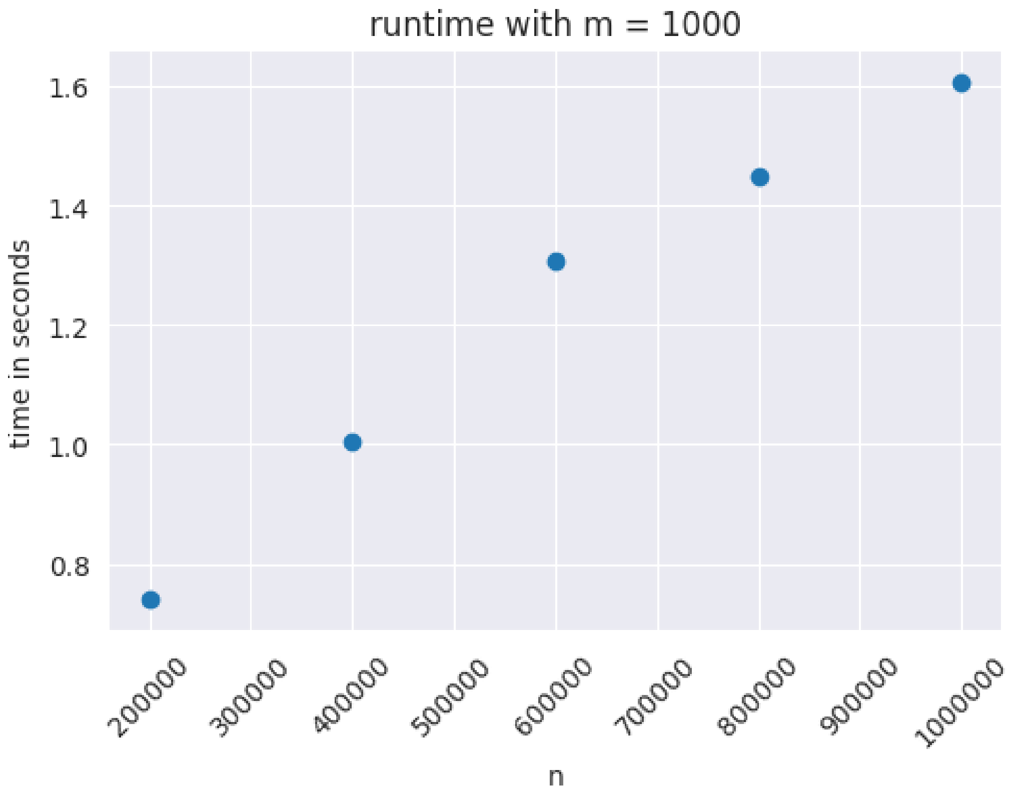
This can be calculated using the mean funtion for the set of random trials. The empirical exptected value is 93 trials to get a collision when the theoretical expected amount is 100

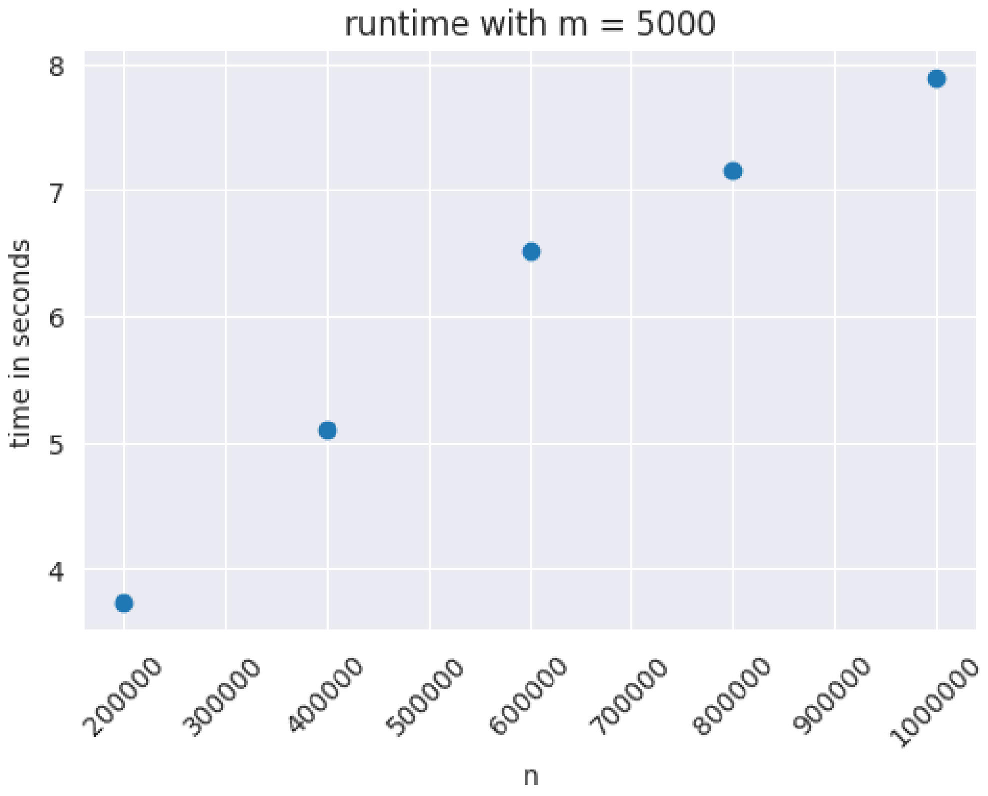
```
print(np.mean(k))
```

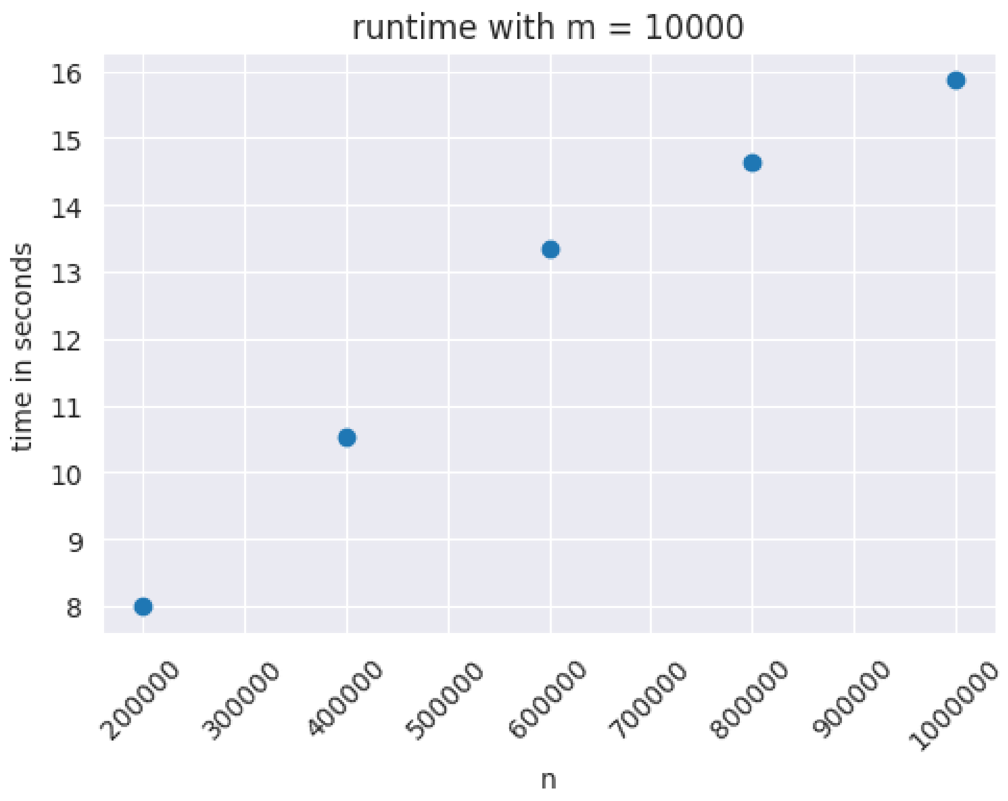
D: (10 points) Describe how you implemented this experiment and how long it took for $m = 300$ trials.

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 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 $m = 10{,}000$.

It took about 0.04 seconds for m =300 and n =5000 to run.
Design of the experiment:
First we need to obstain the number of trials to get a collision. This is done by a while loop, as long as the next random number is not in the set then we append it into the set, else we find a collision and this is when we break the loop and return the number of trials.
Next, we create another loop that run this 300 times and store these k values of collision in a array
Then, k is being sorted from smallest to largest for graphing purposes Run another for loop. For each of these k, we calculate the probability of collision
To calculate run time, import time library and use function time.time()


runtime with m = 1000
time in seconds
0.8
1.0
1.2
1.4
1.6
200000
300000
400000
500000
600000
700000
800000
900000
1000000
n


runtime with m = 5000
time in seconds
4
5
6
7
8
200000
300000
400000
500000
600000
700000
800000
900000
1000000
n



2 Coupon Collectors (35 points)

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

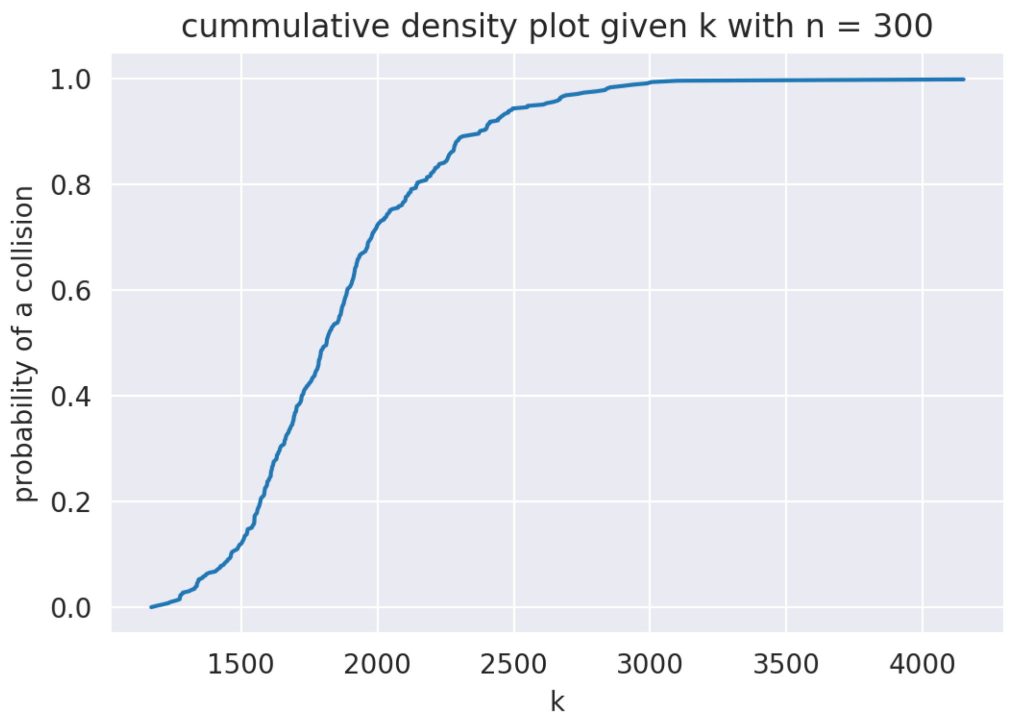
A: (5 points) Generate random numbers in the domain $[n]$ until every value $i \in [n]$ has had one random number equal to i. How many random trials did this take? We will use k to represent this value.

1643 trials to get all types of coupons when the expected amount of trials is 1884

```
import random
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import math
import time

def coupon(size):
    random_numbers_set = set()
    trials = 0
    while True:
        nextRan = random.randint(1,size)
        trials += 1
        if nextRan not in random_numbers_set:
            random_numbers_set.add(nextRan)
            if len(random_numbers_set) == size:# continue until we get all types o
                return trials
    return trials
```

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**.

```
p = []
k = []
trials = 400
prob = 1/trials
while trials > 0:
    nextRand = coupon(n)
    k.append(nextRand)
    trials -= 1
k = sorted(k)
for z in range(len(k)):
    p.append(k.index(k[z])*prob)
print('%d average trials to get all types of coupons'% np.mean(k))
print('when the expected amount of trials is %d'% (n*(0.577+math.log(n))))

sns.set_style("darkgrid")
mpl.rcParams['figure.figsize'] = (6,4)
mpl.rcParams['figure.dpi'] = 200
plt.title("cummulative density plot given k with n = %d" %n)
plt.xlabel("k")
plt.ylabel("probability of a collision")
plt.plot(np.array(k), np.array(p))
plt.show()
```



C: (10 points) Use the above results to calculate the empirical expected value of k.

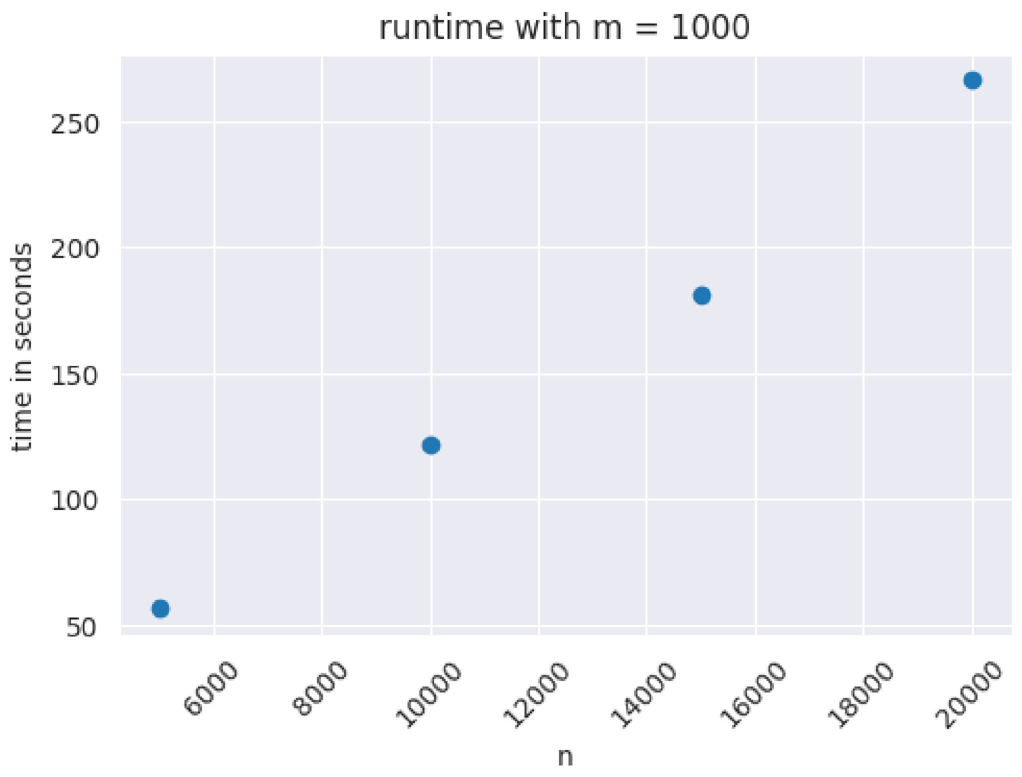
This can be calculated using the mean funtion for the set of trials. The empirical exptected value is 1867 trials to get all types of coupons when the theoretical expected amount of trials is 1884

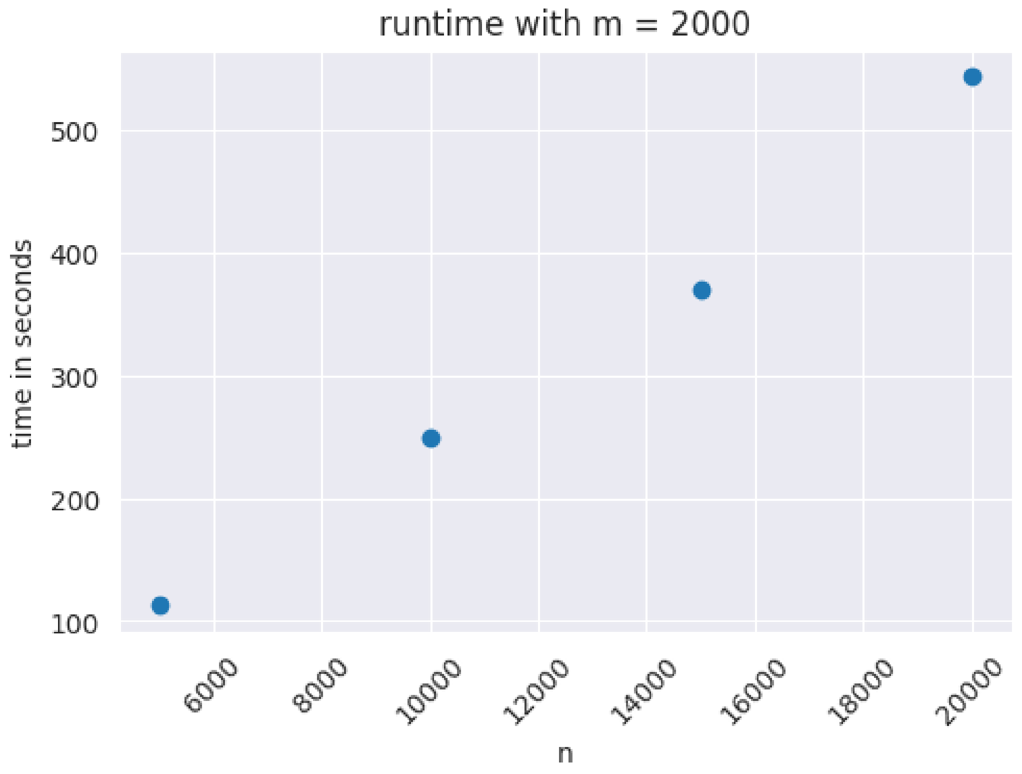
```
print(np.mean(k))
```

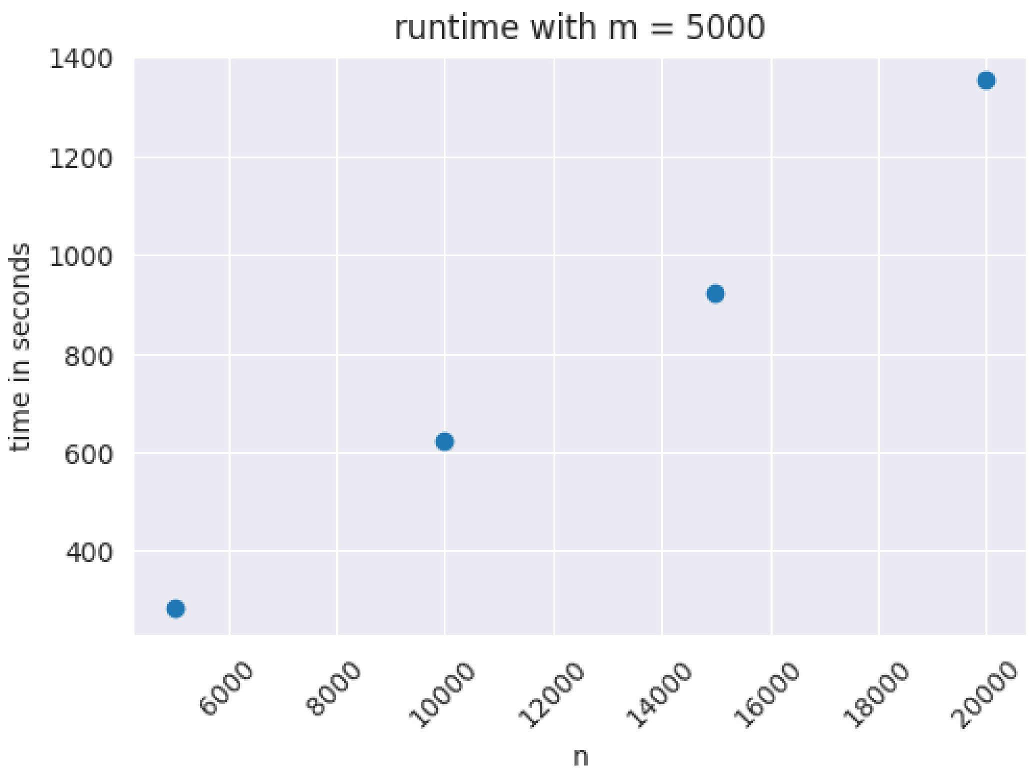
D: (10 points) Describe how you implemented this experiment and how long it took for $n = 300$ and $m = 400$ trials.

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 $m = 5{,}000$.

It took about 0.96 seconds for m =400 and n =300 to run.
Design of the experiment:
Frist, we created a function that returns the number of trials to get all types of coupons.
This is done by using While loop (while true), every loop counts the number of trials until it breaks by the if statement when we get all types of coupons. Inside this while loop, then create an if condition that if the next random number generated are not in the set then add that number into a set.
Inside this if condition, create another if condition to check if the length of the random number set equals to the numbers of all coupons, in this case 300. If it is then stop or returns the number of trials.
Next, we create another loop that run this 400 times and store these k values (number of trials) in a array
Then, k is being sorted from smallest to largest for graphing purposes
Run another for loop. For each of these k, we calculate the probability
To calculate run time, import time library and use function time.time()


runtime with m = 1000
time in seconds
50
100
150
200
250
6000
8000
10000
12000
14000
16000
18000
20000
n


runtime with m = 2000
time in seconds
100
200
300
400
500
6000
8000
10000
12000
14000
16000
18000
20000
n



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**?

The probability that any two have a collision is $\frac{1}{n}$

To measure that at least one pair have a collision, it is easier to measure the probability that no pair has a collision. In general, there are $\binom{k}{2}$ **pairs.**

So, P(no pair is the same)= $(1 - \frac{1}{n})^{\binom{k}{2}}$

And, P(at least one collision) = $1 - (1 - \frac{1}{n})^{\binom{k}{2}}$

If we have n equi-probability random events, then expect after about $k = \sqrt{2n} = \sqrt{2 \times 5000} = 100$ **trials to get a collision**

When k =100, P(at least one collision) = $1 - (1 - \frac{1}{5000})^{\binom{100}{2}} \approx .63$

From 1.C, we got average of trials is 93**, so P(at least one collision) =** $1 - (1 - \frac{1}{5000})^{\binom{93}{2}} \approx .58$**. Hence, the results we got from 1C makes sense**

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.]
How does this compare to your results from **2.C**?

Let r_i be the expected number of trials to receive exactly i distinct coupons.
Let $r_0 = 0$ and set $t_i = r_i - r_{i-1}$ be the exptected number of trials between getting $i - 1$ distinct coupons and i distinct coupons
The exptected number of trials to get all coupons is $T = \sum_{i=1}^{n} t_i$
Let p_i be the probability that we get a new coupon after having $i - 1$ distinct coupons.
Thus, $t_i = \frac{1}{p_1}$. And $p_i = \frac{(n-i+1)}{n}$

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

$k = T = n \sum_{i=1}^{n} \frac{1}{i} \approx n(\gamma + \ln n) = 300(0.577 + \ln 300 = 1884$ **(Harmonic Number with $\gamma = 0.577$)**
From 2.C, we got the average trials of 1867, which is close to our expectation, 1884. So the result makes sense

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 $\mathbf{E}[f_i] = k/n$. Let $\mu = \max_{i\in[n]} f_i/k$.

Consider some parameter $\varepsilon \in (0, 1)$. As a function of parameter ε, how large does k need to be for $\mathbf{Pr}[|\mu - 1/n| \geq \varepsilon] \leq 0.05$? That is, how large does k need to be for *all* counts to be within $(\varepsilon \cdot 100)\%$ 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 $\mathbf{Pr}[|\mu - 1/n| \geq \varepsilon] \leq 0.005$ (that is, only 0.005 probability of exceeding ε error)?

[Make sure to show your work.]