EE325 Assignment 2

Anoushka Dey 210010010
Savaliya Abhishek 21D070065
Ritesh Bahl 21D070057

September 6, 2022

1 Question 1

The values of p given here are 10, 20, 50 and 75. First a function was defined to calculate probability as $\frac{\binom{m}{p}\cdot\binom{n-m}{m-p}}{\binom{n}{m}}$.

The definitions of n, m and p are as those specified in the question. The aim here is to predict a value of n, that is, find an estimate for the actual number of fishes in the lake.

The probabilities were first plotted taking various values of n into consideration. The value of n for which the probability turned out to be maximum was defined to be our best estimate of the actual number of fishes for various values of p.
This is the code:

```
import numpy as np
import matplotlib.pyplot as plt
import random as random
from math import comb
def probcalc(n,m,p):
    return (comb(m,p)*(comb((n-m),(m-p))))/(comb(n,m))
p=[10,20,50,75]
def plot_prob(m,p,k):
    x=[i for i in range(m,3000)]
    y=[probcalc(i,m,p) for i in x]
    nbest=y[0]
    for i in x:
        if(y[x.index(i)]>nbest):
            nbest=y[x.index(i)]
            n=i
    if k==1:
        plt.plot(x,y,markersize=1)
        plt.plot(n,nbest,'ro')
        plt.xlabel('No. of Fishes')
        plt.ylabel('Probability of finding such no. of fishes')
```

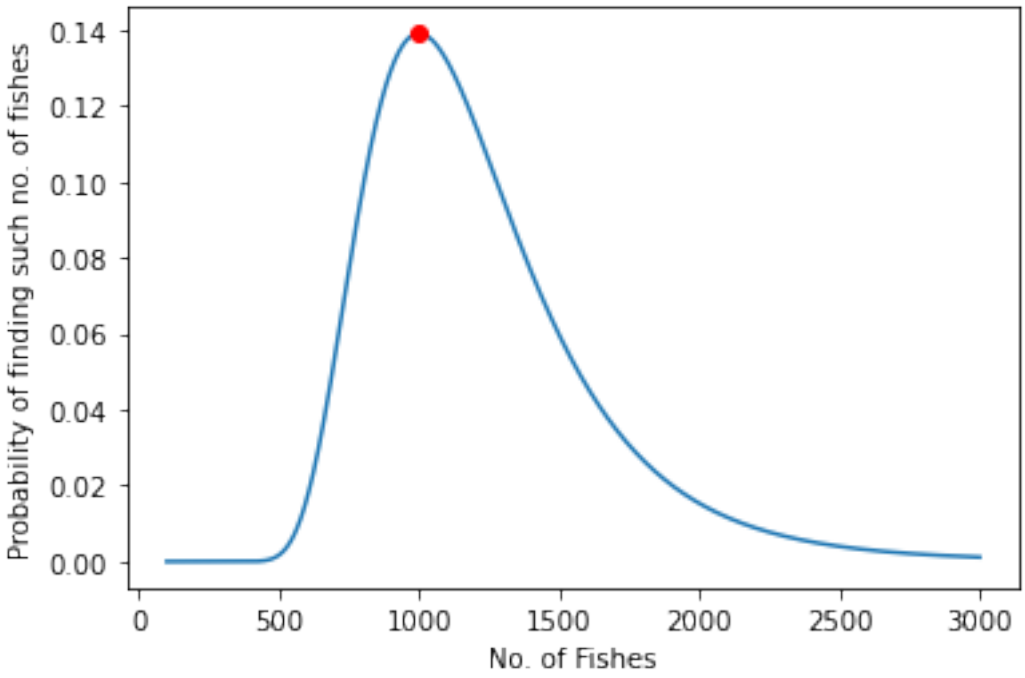

Figure 1: n=999

```
    return
plot_prob(100,10,1)
plot_prob(100,20,1)
plot_prob(100,50,1)
plot_prob(100,75,1)
```

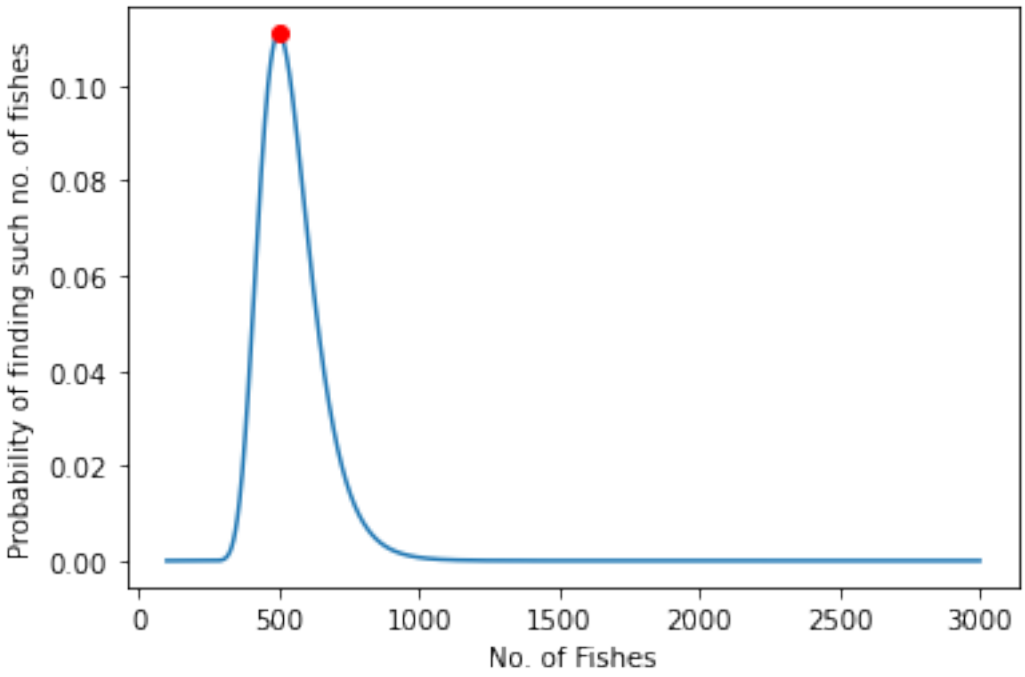

Figure 2: n=499

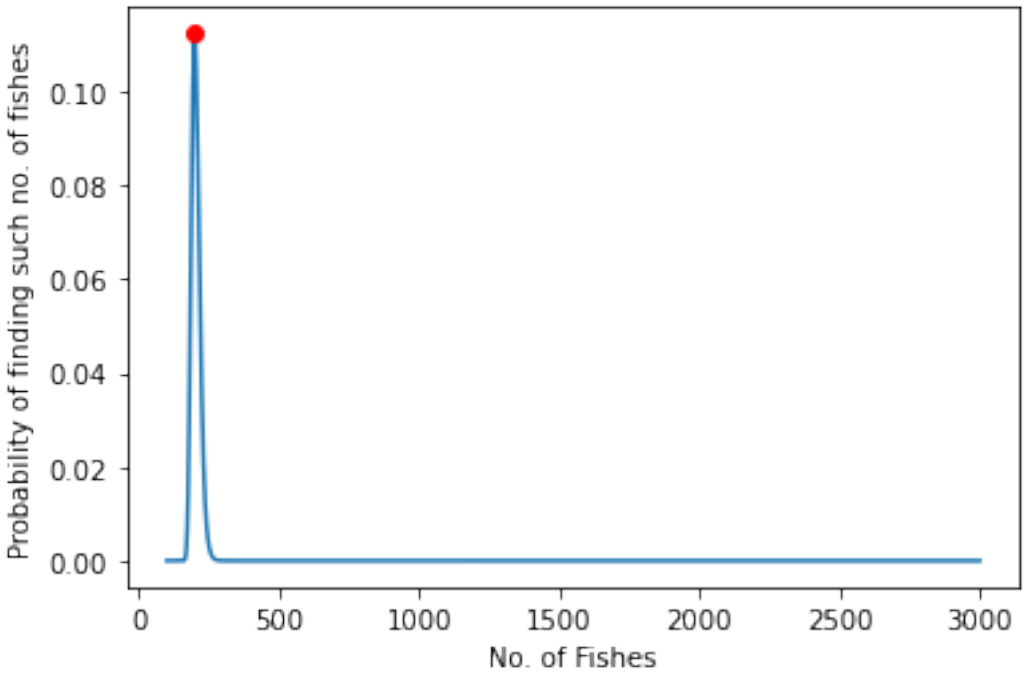

Figure 3: n=199

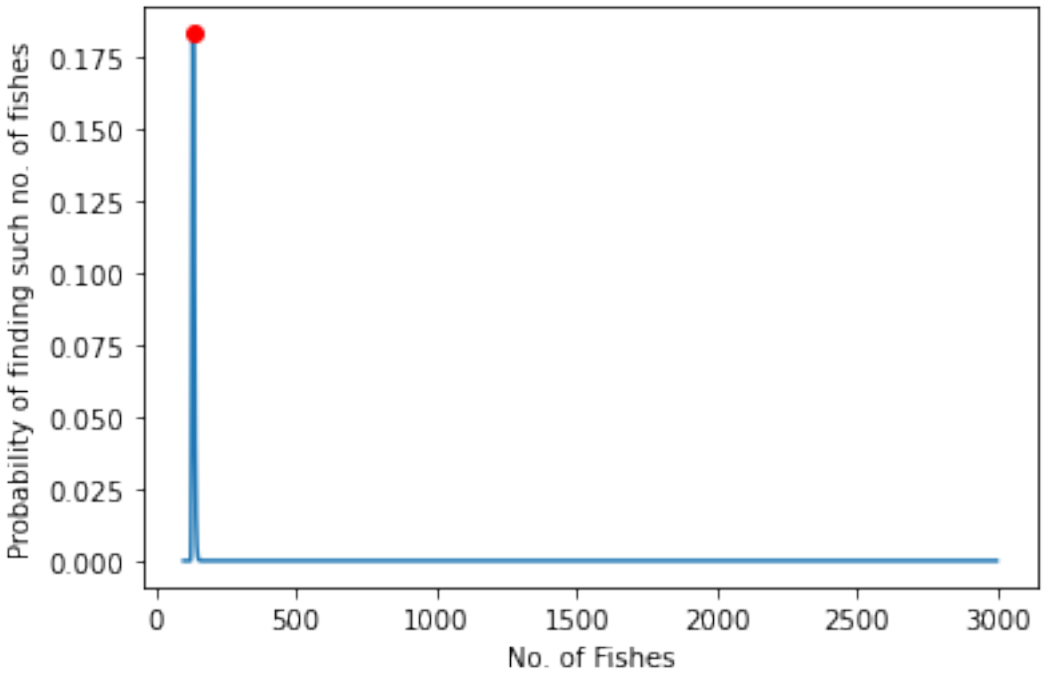

Figure 4: n=133

2 Question 2

In this question, we have first created an array of size 2000 containing only zeroes. After this we have randomly selected 100 locations and changed the value at that index to 1. Subsequently, we have followed the same procedure as above 500 times for getting a value of n for each set of random sampling of 100 samples of fishes. The error $n - 1000$ has also been plotted.
This is the code:

```
import numpy as np
import matplotlib.pyplot as plt
import random as random
nz=np.zeros(2000)
idx=[]
for i in range(0,100):
    idx.append(random.randint(0,1999))
for i in idx:
    nz[i]=1
err=[]
id=[]
for i in range(0,500):
    id.append(random.sample(range(0,2000),100))
def probcalc(n,m,p):
    return (comb(m,p)*(comb((n-m),(m-p))))/(comb(n,m))
def plot_prob(m,p,k):
    x=[i for i in range(m,3000)]
    y=[probcalc(i,m,p) for i in x]
    nbest=y[0]
    for i in x:
        if(y[x.index(i)]>nbest):
            nbest=y[x.index(i)]
            n=i
    if k==1:
        plt.plot(x,y,markersize=1)
        plt.plot(n,nbest,'ro')
        plt.xlabel('No. of Fishes')
        plt.ylabel('Probability of finding such no. of fishes')
    return
for i in range(0,500):
    c=0
    for x in id[i]:
        if nz[x]==1:
            c+=1
    n=plot_prob(100,c,0)
    err.append((n-1000))
plt.plot(err,'ro')
```

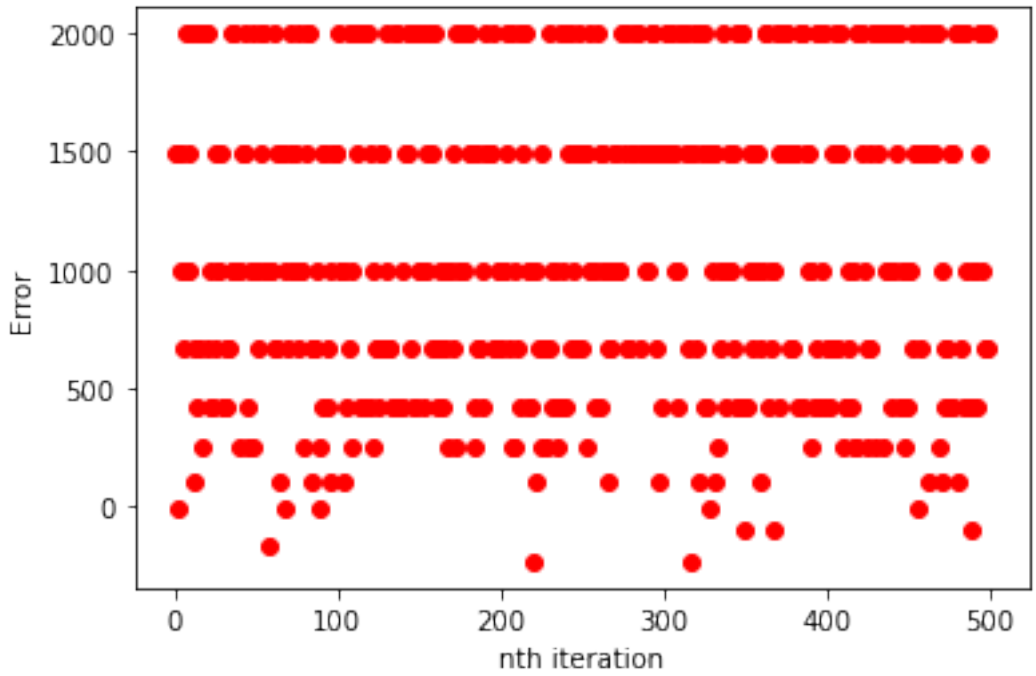

Figure 5: Error Plot

```
plt.xlabel('nth iteration')
plt.ylabel('Error')
np.mean(err)
np.var(err)
```

Here mean error= 1117.52 and variance= 404811.9816

3 Question 3

In this question, we have to find p(n) which is

$$p(n) = \frac{Numberoftimestepsinwhichwehavenpacketsinthequeue}{Totalnumberoftimesteps}$$

So I defined an array called queue and ran a for loop which simulates the process for given no. of time steps.

```
import matplotlib.pyplot as plt;
import numpy as np;
import random;
```

```
no_of_time_steps = 1000000
"""
Defining Queue
Although Queue has infinite memory, to define it we need to define it's array size
Since max no. of elements in queue can not be greater than no. of time steps
(because at each time step max one packet can arrive);
queue length can be maximum upto no. of time steps.
"""
Queue_length = no_of_time_steps
Queue = np.zeros(Queue_length)
n = np.arange(0, 51) #n is no. of packets. We will plot this array on x axis
n_current=0          #no. of packets in current queue
Queue_end=0          #End point of current Queue
Queue_start=0
"""
Untill End point Of current queue,
we will take all all elements of queue as 1 and after end point we will take all points 0
"""
no_of_time_steps_at_which_there_are_n_samples_in_queue =np.zeros(51)
time_steps = np.arange(1, no_of_time_steps + 1)
sum_of_no_of_packets_in_each_time_step = 0
#This will be numerator while finding average value

arrival_array = [1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
transmission_array = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

for i in range (no_of_time_steps):
  if ( random.choice(arrival_array)== 1 ):
    if(n_current == 0):
        Queue[Queue_end]=1
    else:
        Queue_end=(Queue_end +1) % Queue_length
        Queue[Queue_end]=1
    n_current +=1
  if ( random.choice(transmission_array)==1):
     if(n_current != 0):
        Queue[Queue_start]=0
        Queue_start=(Queue_start +1) % Queue_length
        n_current -=1
  no_of_time_steps_at_which_there_are_n_samples_in_queue[n_current] += 1
  sum_of_no_of_packets_in_each_time_step += n_current

#plotting
plt.plot( n , no_of_time_steps_at_which_there_are_n_samples_in_queue/ no_of_time_steps)
plt.title("Plot of P(n)")
```

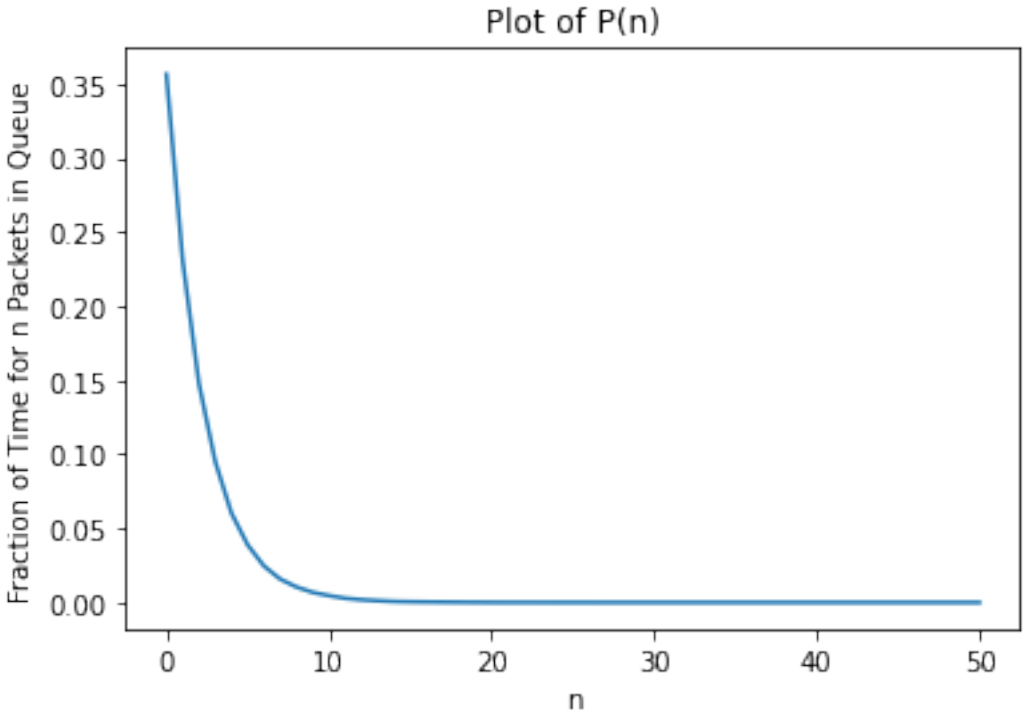

Figure 6: Plot for p(n)

```
plt.xlabel("n")
plt.ylabel("Fraction of Time for n Packets in Queue")

time_avg = sum_of_no_of_packets_in_each_time_step/no_of_time_steps
print(time_avg)
```

Time average of number of packets in the memory is coming out to be 1.8

4 Question 4

Now we have to simulate previous question for 10000 queues in 100000 time steps. Also in this quesion p(n) is

$$p(n) = \frac{Numberofqueuesinwhichwehavenpacketsinthequeue}{Totalnumberoftimesteps}$$

```
import matplotlib.pyplot as plt;
import numpy as np;
import random;
```

```
no_of_queues = 10000
no_of_time_steps = 100000

Queue_length = no_of_time_steps
Queue = np.zeros(Queue_length)
n = np.arange(0, 51) #n is no. of packets. We will plot this array on x axis
n_current=np.arange(0,no_of_queues)          #no. of packets in current queue
Queue_end=0          #End point of current Queue
Queue_start=0

no_of_time_steps_at_which_there_are_n_samples_in_queue =np.zeros(51)
time_steps = np.arange(1, no_of_time_steps + 1)
sum_of_no_of_packets_in_each_time_step = 0
no_of_queues_that_have_n_packets=np.zeros(51)

arrival_array = [1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
transmission_array = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
sum=0

for j in range (no_of_queues):
 for i in range (no_of_time_steps):
  if ( random.choice(arrival_array)== 1 ):
    if(n_current[j] == 0):
        Queue[Queue_end]=1
    else:
        Queue_end=(Queue_end +1) % Queue_length
        Queue[Queue_end]=1
    n_current[j] +=1
  if ( random.choice(transmission_array)==1):
     if(n_current[j] != 0):
        Queue[Queue_start]=0
        Queue_start=(Queue_start +1) % Queue_length
        n_current[j] -=1
 sum +=n_current[j]
 no_of_queues_that_have_n_packets[n_current[j]] += 1
sample_avg = sum/no_of_queues
print(sample_avg)
x_axis = np.arange(0,51)
plt.plot(x_axis, no_of_queues_that_have_n_packets/no_of_queues )
```

Sample Average came out to be 1.83

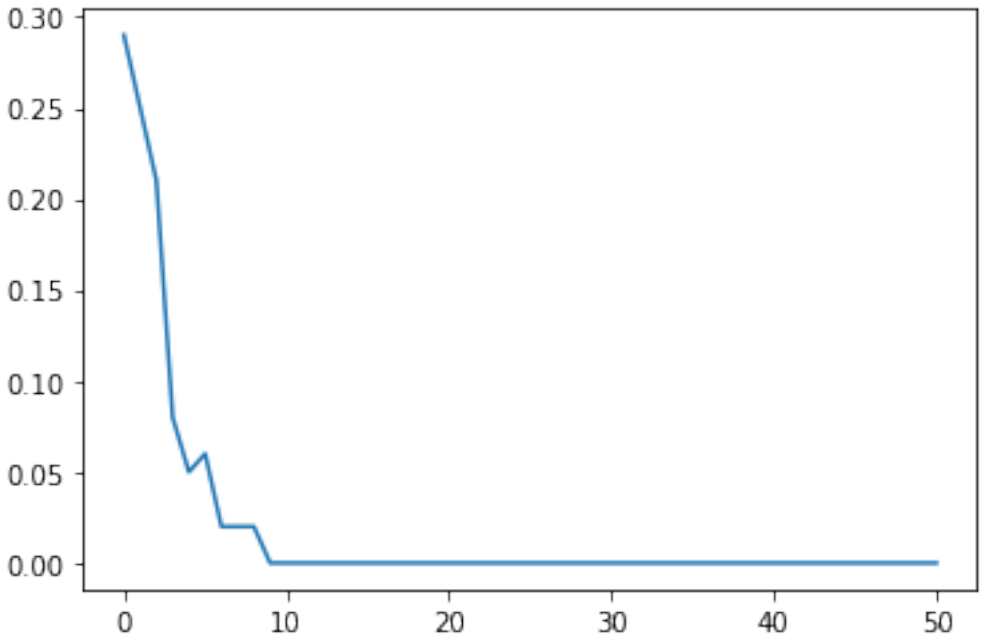

Figure 7: Plot for p(n)