EE 325: PROBABILITY AND RANDOM PROCESSES

Under the guidance of PROF. D. MANJUNATH

Team details:

- **Abhijat Bharadwaj (210020002)**
- **Aryan Shah (21D170008)**
- **Debasish Panda (21D070021)**

We as a whole team have put necessary inputs at various stages during the assignment. We have discussed the questions, discussed upon various ways to approach the problem and came up with an answer/ way that we unifying agreed upon.

We have followed a procedure overall, we have written a code for the question, followed by some explanation and theoretical answers to the questions, as per asked. Also, we have tried including necessary graphs and plots for the data, to get a better visualization of what we want to say.

Problem 1

1(a) :

Without any loss of generality, let us assume that $p_C > p_B > p_A$. Hence, after N_1 trials, if we pick up any coin other than C, it would be the wrong choice. Let i, j, and k be the number of heads appearing from $m = N_1/3$ tosses of A, B, and C, respectively. Thus, the probabiity of picking up the correct coin can be expressed as:

P(C) = P(k>=j) P(k>=i), where,

P(k>=j) = $\sum_{j=0}^{m} \sum_{k=j}^{m} \left({}^{m}C_{j} \, {}^{m}C_{k} \, p_B^{j} p_C^{k} \left(1-p_B\right)^{m-j} \left(1-p_C\right)^{m-k} \right)$

P(k>=i) = $\sum_{i=0}^{m} \sum_{k=i}^{m} \left({}^{m}C_{i} \, {}^{m}C_{k} \, p_A^{i} p_C^{k} \left(1-p_A\right)^{m-i} \left(1-p_C\right)^{m-k} \right)$

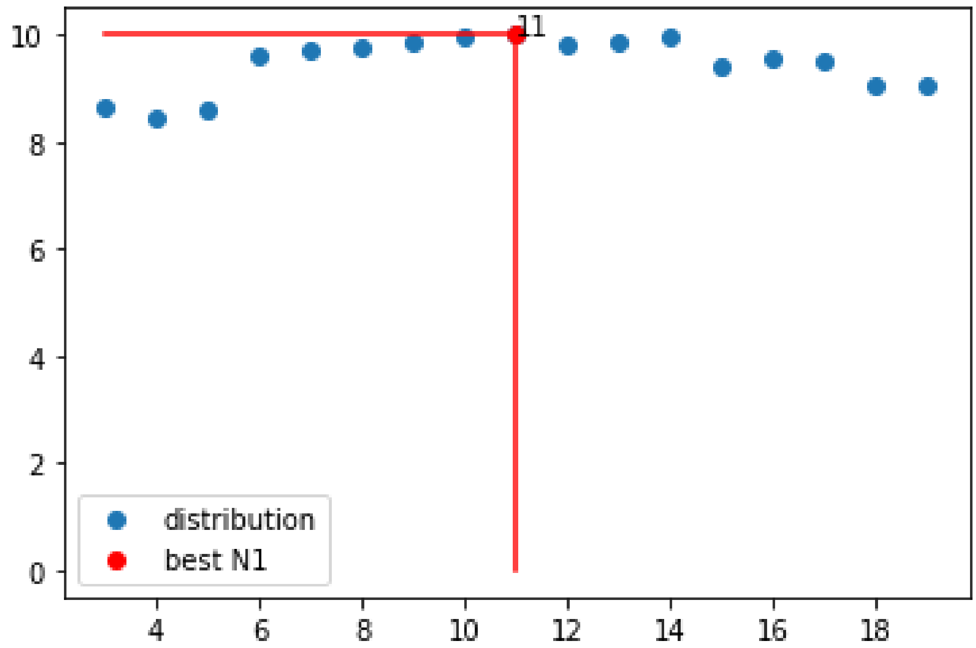
Thus the probability of picking up the wrong coin(either of A or B) can be expressed as, P(wrong) = 1-P(C).

1(b).(i) :

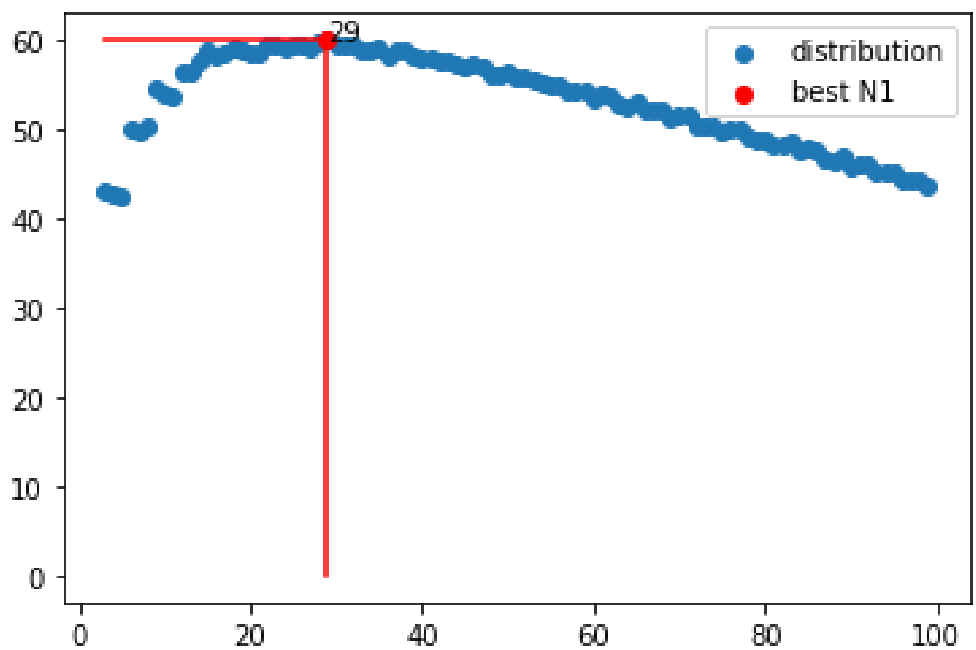
```
from cProfile import label
from math import ceil
from turtle import color
import matplotlib.pyplot as plt
from statistics import mean
import random
h=0
hA,hB,hC,oA,oB,oC=0,0,0,0,0,0
def toss(p):
    arr=[1]*int(p*100)+[0]*int((1-p)*100)
    return random.choice(arr)
N=[20,100,1000,5000]
p_abc=[(0.2,0.4,0.7),(0.45,0.5,0.58)]      #pA,pB,pC
for (pA,pB,pC) in p_abc:
    for n in N:
        R_arr=[]
        for N1 in range (3,n):
            heads=[]
            for j in range(1000):
                hA,hB,hC,h=0,0,0,0
                ni=N1//3
                for i in range(ni):
                    t=toss(pA)
                    h+=t
                    hA+=t
                for i in range(ni):
                    t=toss(pB)
                    h+=t
```

```
                hB+=t
            for i in range(ni):
                t=toss(pC)
                h+=t
                hC+=t
            for i in range(n-3*ni):
                if (hA>=hB)and(hA>=hC):
                    t=toss(pA)
                    h+=t
                    #hA+=t
                elif (hB>=hA)and(hB>=hC):
                    t=toss(pB)
                    h+=t
                    #hB+=t
                elif (hC>=hB)and(hC>=hA):
                    t=toss(pC)
                    h+=t
                    #hC+=t
            heads.append(h)
        R=mean(heads)
        R_arr.append(R)
    plt.scatter(range(3,n),R_arr,label="distribution")

    plt.scatter([R_arr.index(max(R_arr))+3],
[max(R_arr)],color='red',label="best N1")
    plt.annotate(xy=(R_arr.index(max(R_arr))+3 , max(R_arr)),
s=R_arr.index(max(R_arr))+3)
    plt.plot(range(3,R_arr.index(max(R_arr))+4),
[max(R_arr)]*(R_arr.index(max(R_arr))+1),color='red')
    plt.plot([R_arr.index(max(R_arr))
+3]*ceil(max(R_arr)),range(0,ceil(max(R_arr))),color='red')
    plt.legend()
    plt.show()
```

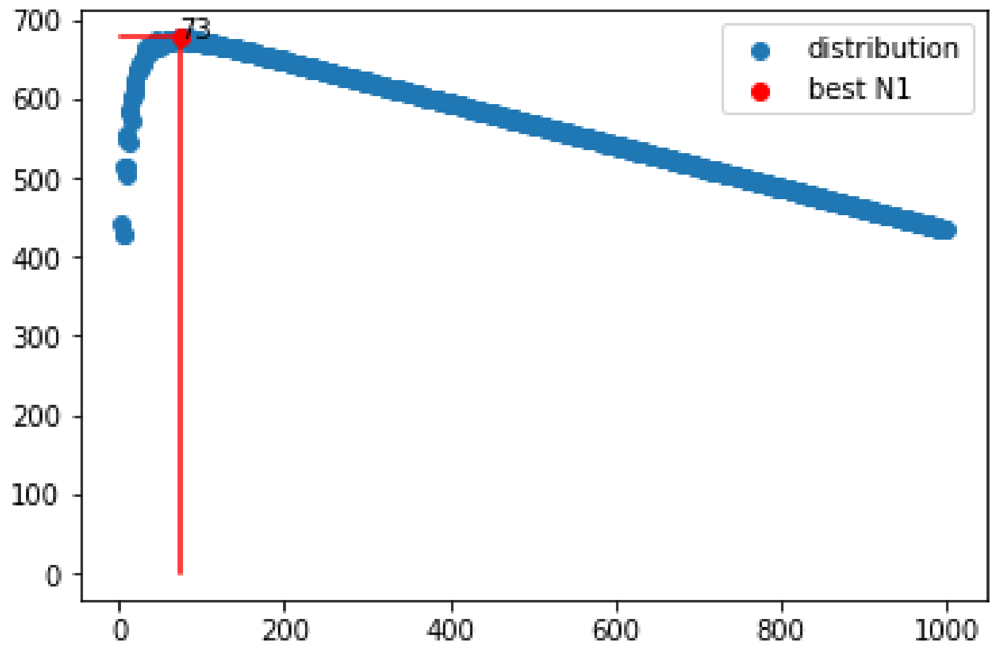
1(a).(i).(1) : N = 20



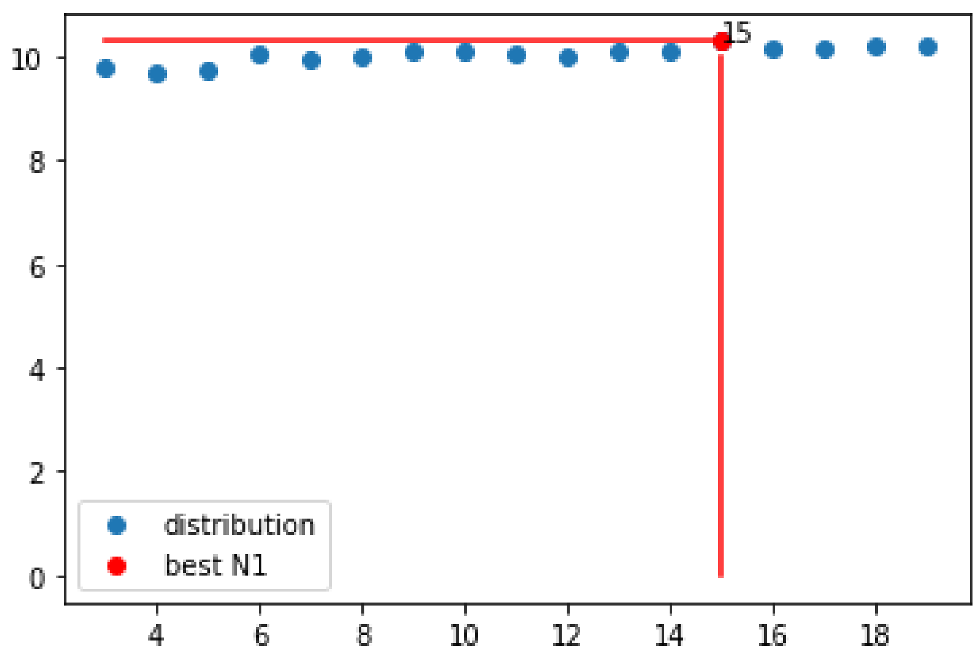
1(a).(i).(2) : N = 100



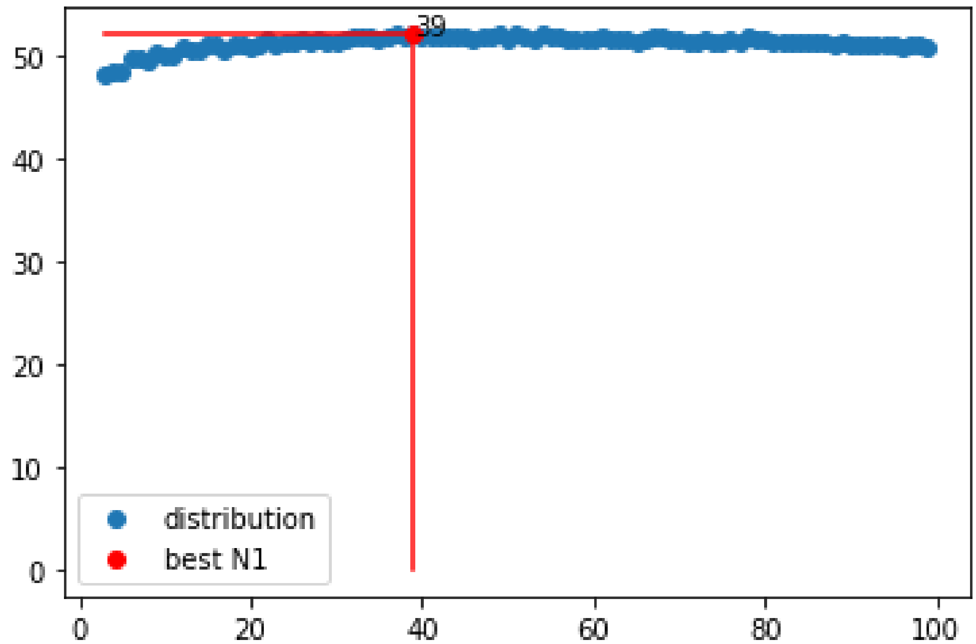
1(a).(i).(3) : N = 1000



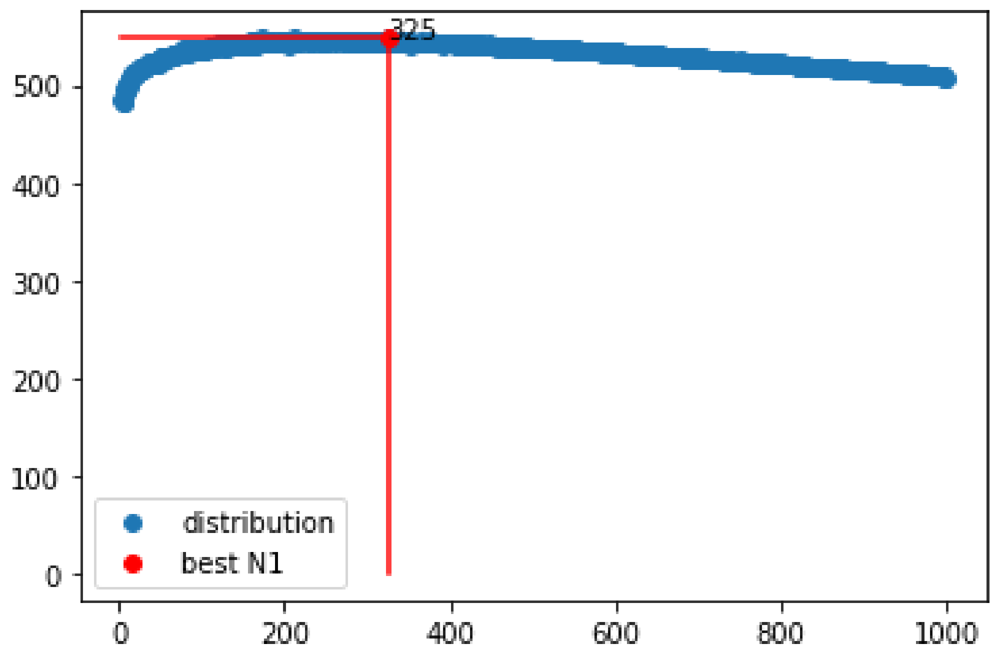
1(a).(ii).(1) : N = 20



1(a).(ii).(2) : N = 100



1(a).(ii).(3) : N = 1000



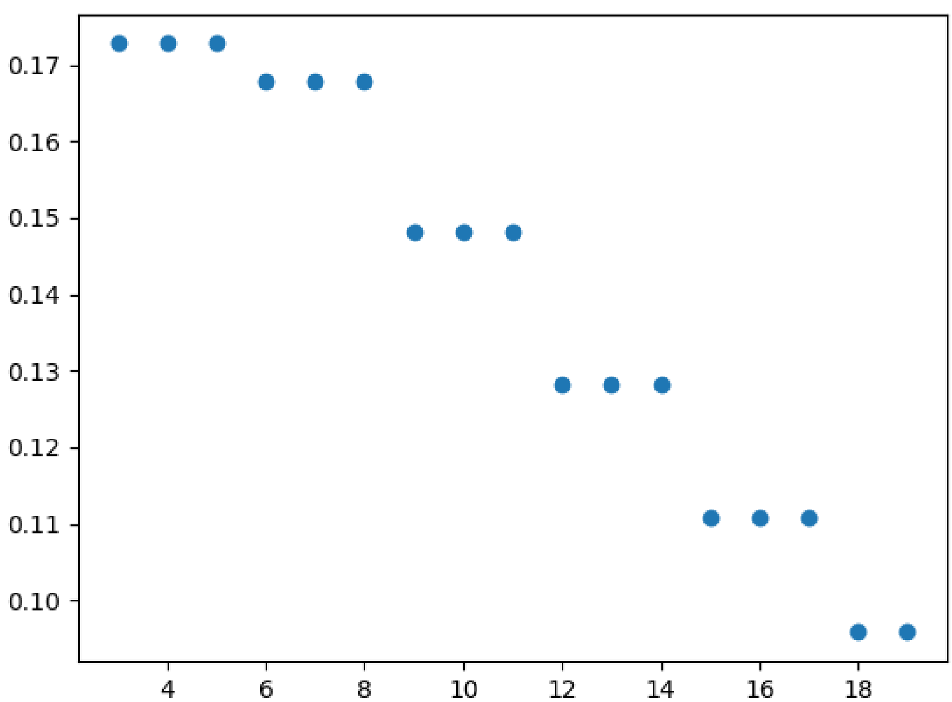
1(b).(ii) :

Thoretical probability :

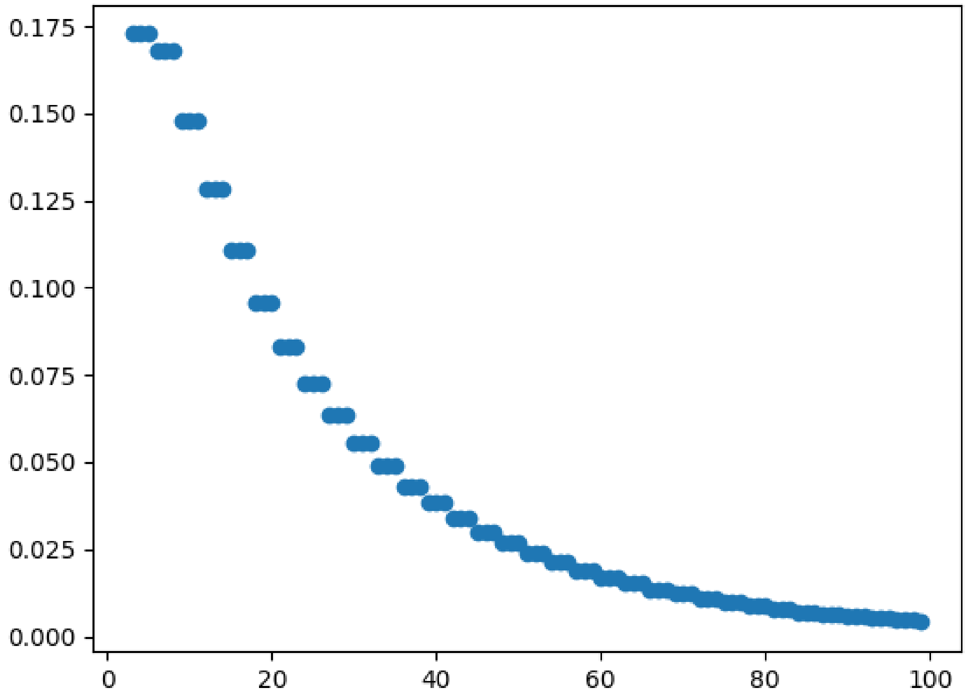
```
import matplotlib.pyplot as plt
from statistics import mean
import random
from math import comb
```

```
N=[20,100,1000,5000]

p_abc=[(0.2,0.4,0.7),(0.45,0.5,0.58)]    #pA,pB,pC

Pca,Pcb=0,0 #C>=A,C>=B

for (pA,pB,pC) in p_abc:
    for n in N:
        pWRONG=[]
        for N1 in range (3,n):
            m=N1//3
            Pca,Pcb=0,0
            for j in range(m+1):
                for k in range(j,m+1):
                    Pca+=comb(m,j)*comb(m,k)*(pA**j)*(pC**k)*((1-pA)**(m-j))*((1-pC)**(m-k))
                    Pcb+=comb(m,j)*comb(m,k)*(pB**j)*(pC**k)*((1-pB)**(m-j))*((1-pC)**(m-k))

            pWRONG.append(1-(Pca*Pcb))
        plt.scatter(range(3,n),pWRONG)
        plt.show()
```

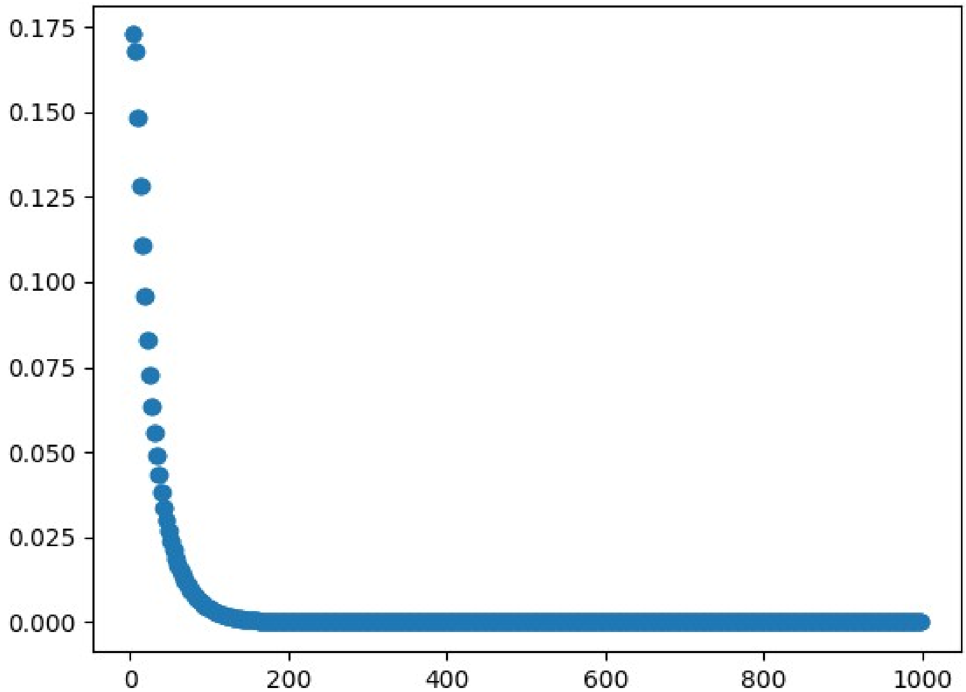
1(b).(i).(1) : N = 20



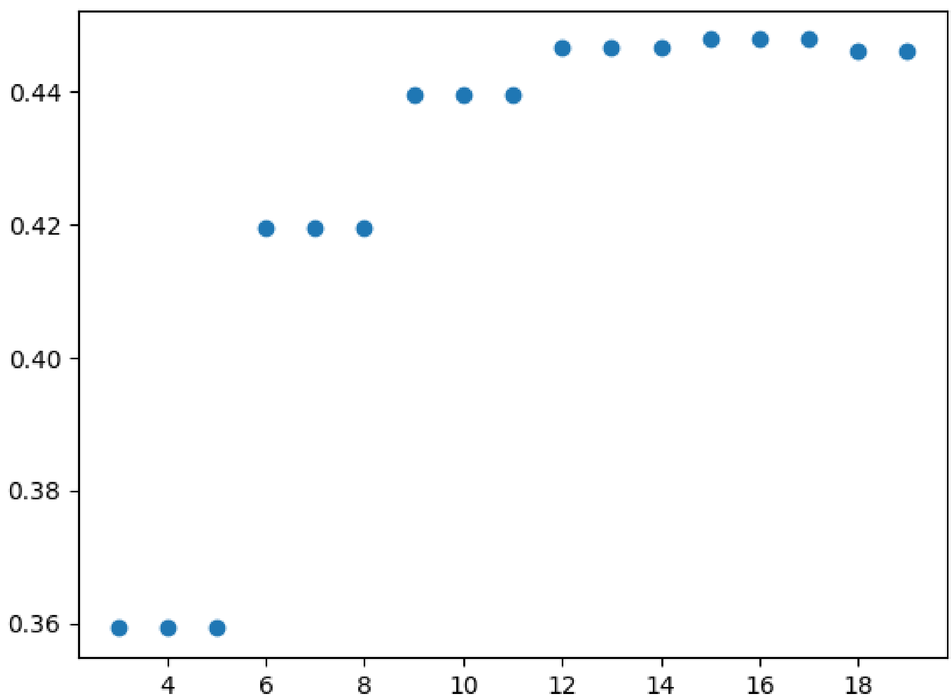
1(b).(i).(2) : N = 100



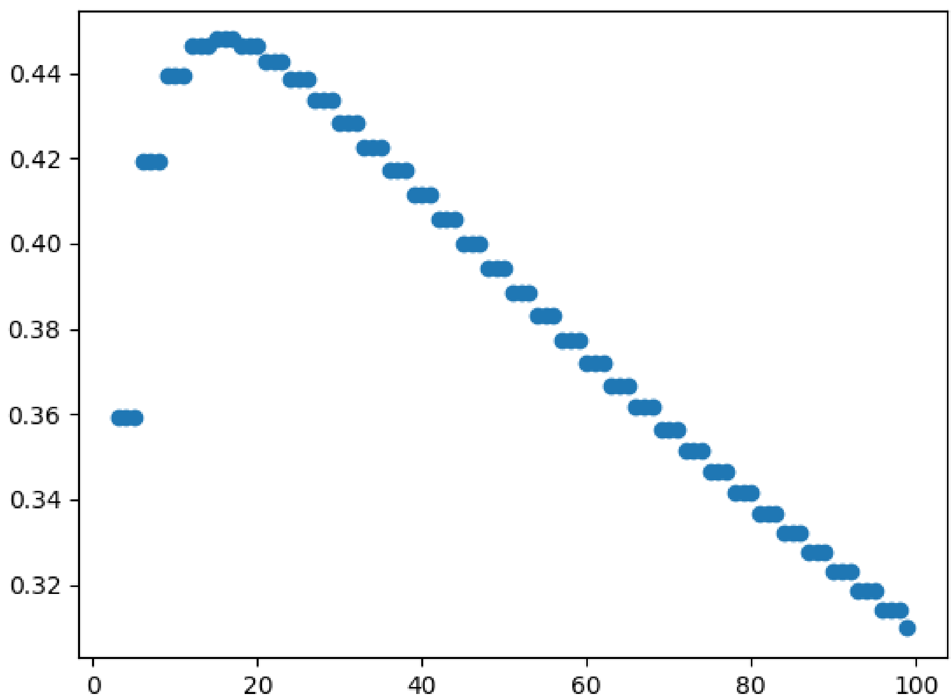
1(b).(i).(3) : N = 1000



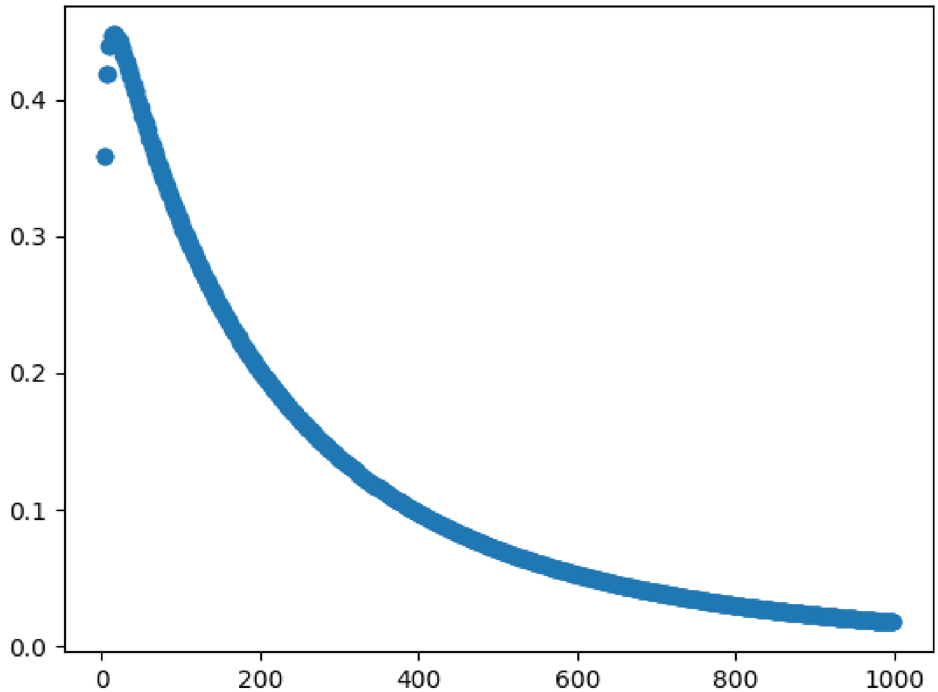
1(b).(ii).(1) : N = 20



1(b).(ii).(2) : N = 100



1(b).(ii).(3) : N = 1000

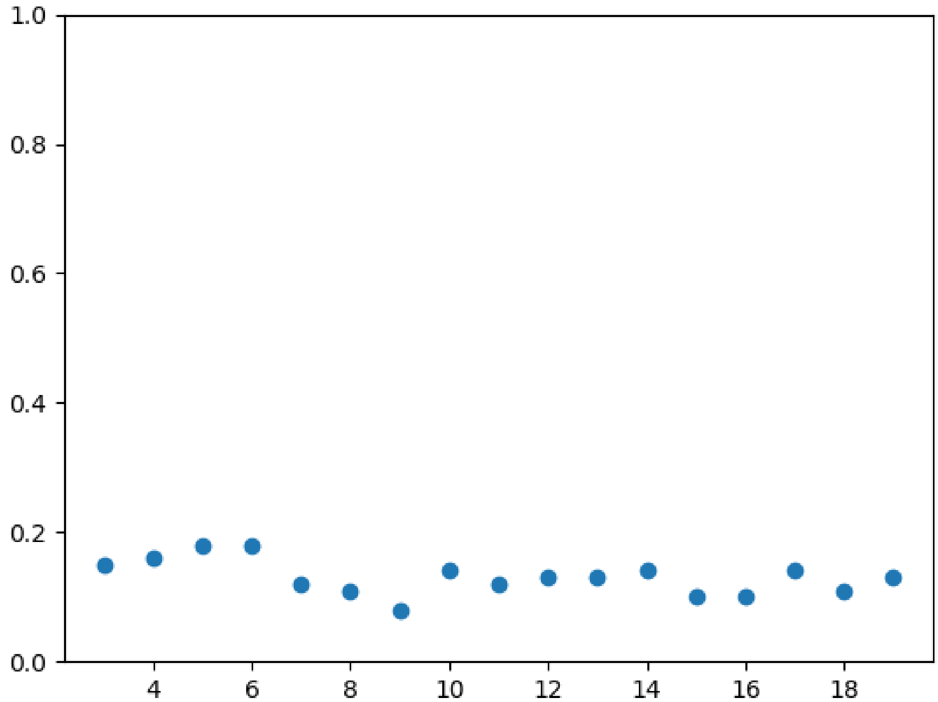


Empirical probability :

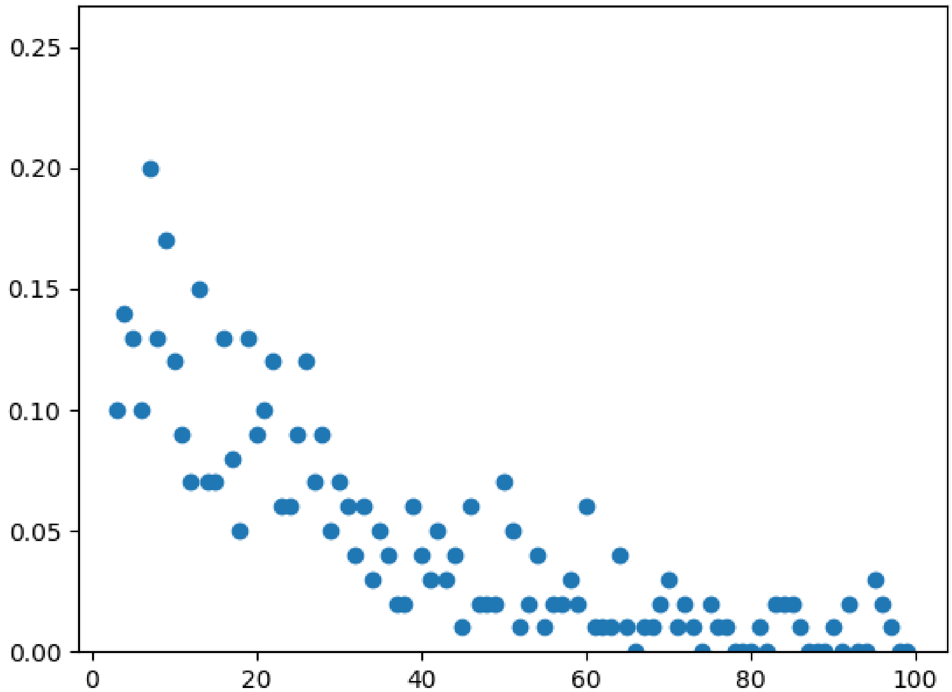
```
from cProfile import label
from math import ceil
from turtle import color
import matplotlib.pyplot as plt
from statistics import mean
import random
plt.ylim([0,1])

h=0
hA,hB,hC,oA,oB,oC=0,0,0,0,0,0
def toss(p):
    arr=[1]*int(p*100)+[0]*int((1-p)*100)
    return random.choice(arr)
N=[20,100,1000,5000]
p_abc=[(0.2,0.4,0.7),(0.45,0.5,0.58)]    #pA,pB,pC
for (pA,pB,pC) in p_abc:
    for n in N:
        W=[]
        for N1 in range (3,n):
            w=0
            for i in range(100):
                hA,hB,hC,h=0,0,0,0
                ni=N1//3
```

```
            for i in range(ni):
                t=toss(pA)
                h+=t
                hA+=t

            for i in range(ni):
                t=toss(pB)
                h+=t
                hB+=t

            for i in range(ni):
                t=toss(pC)
                h+=t
                hC+=t

            if (hC<hA)or(hC<hB):
                w+=1
            else:
                w+=0

        w/=100
        W.append(w)
    plt.scatter(range(3,n),W)
    plt.show()
```

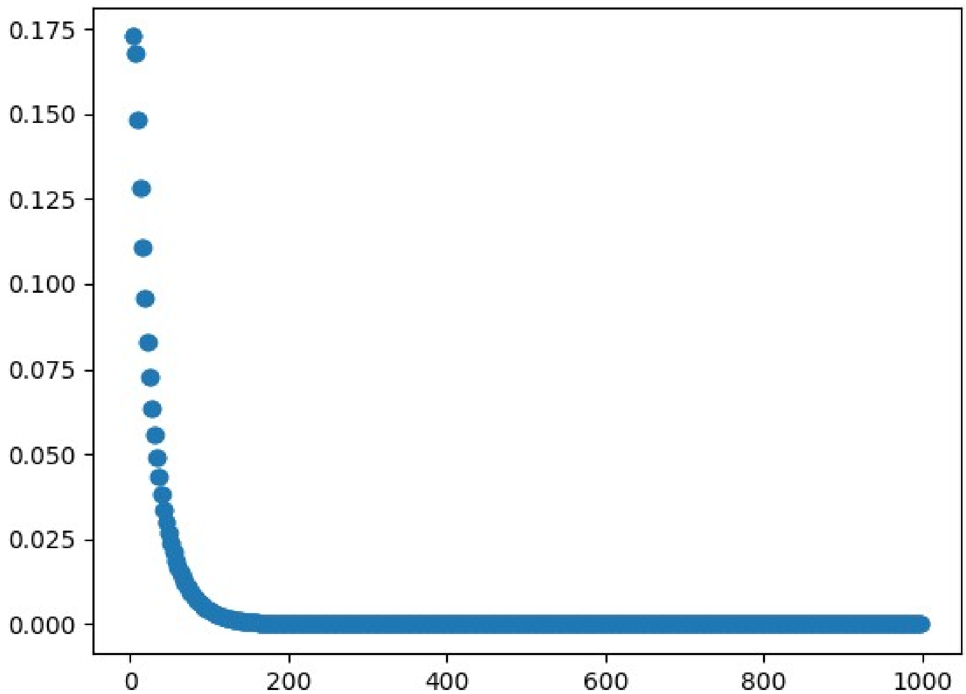
1(b).(i).(1) : N = 20



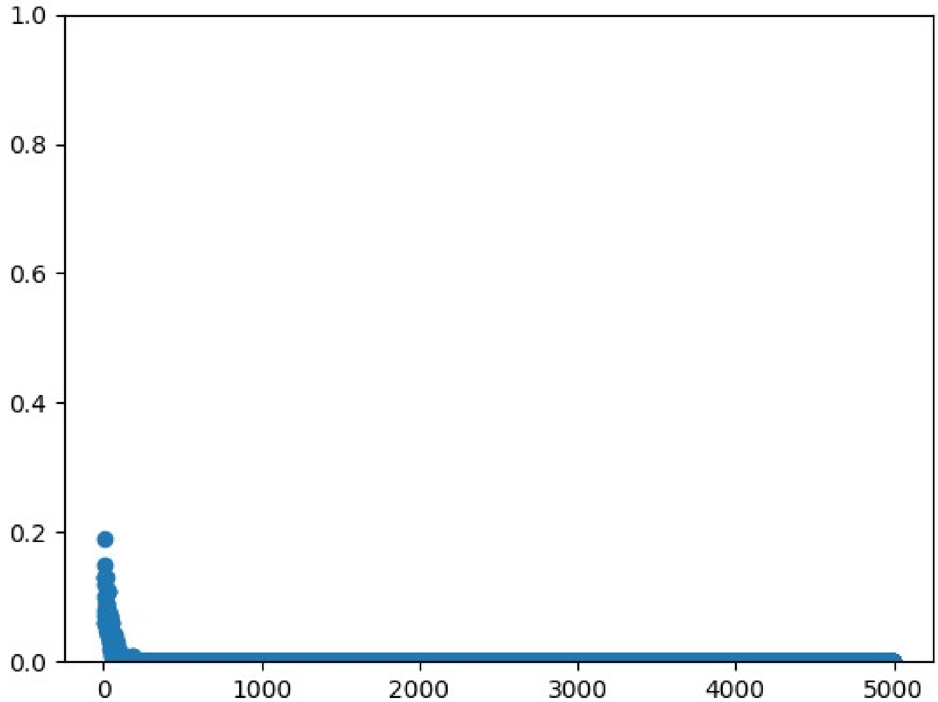
1(b).(i).(2) : N = 100



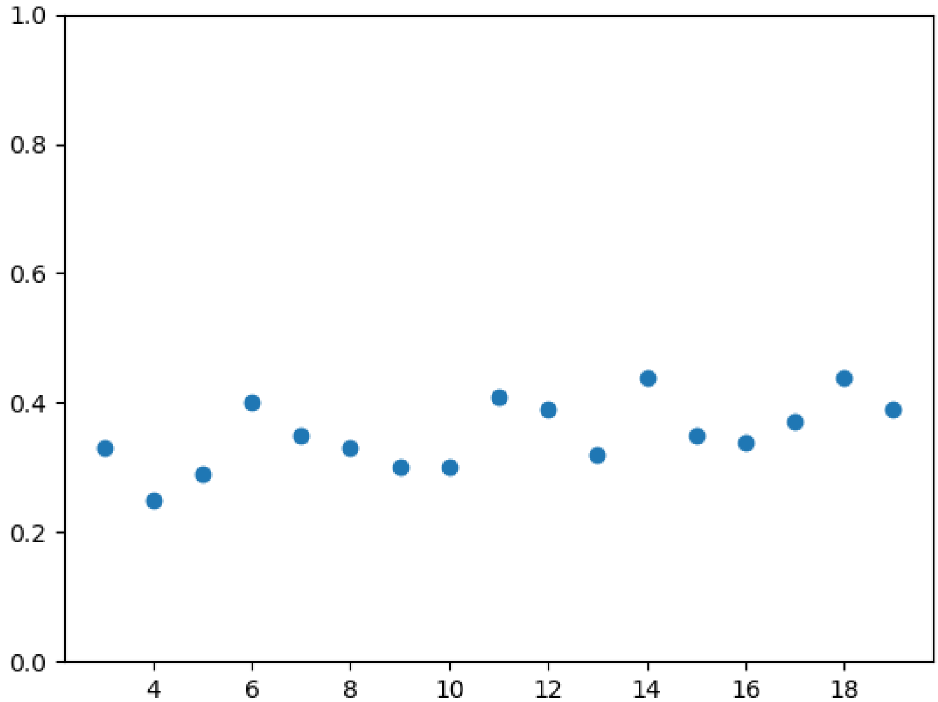
1(b).(i).(3) : N = 1000



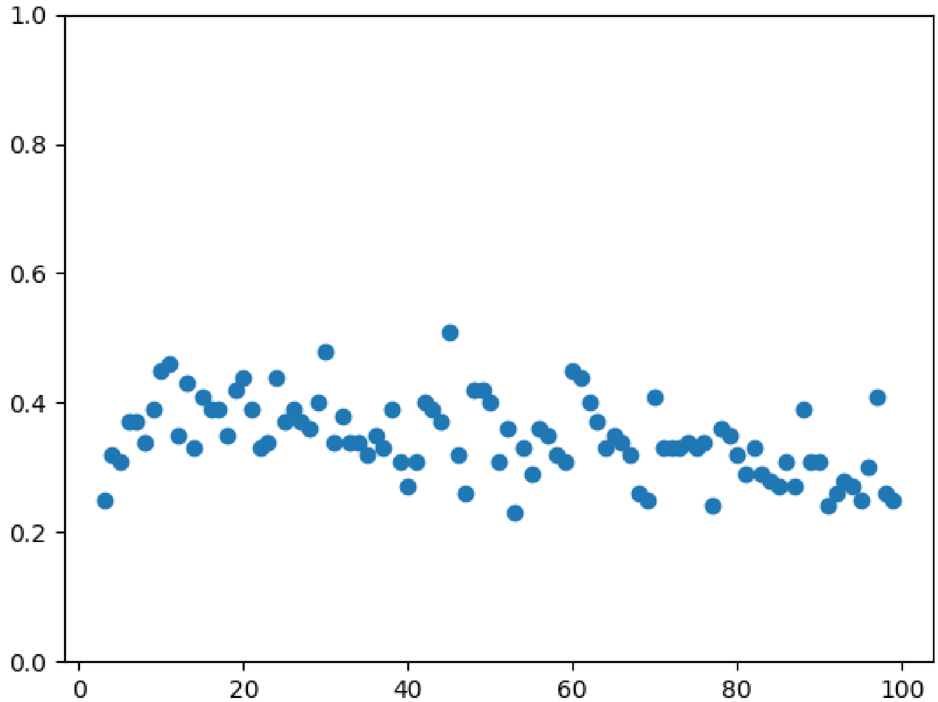
1(b).(i).(4) : N = 5000



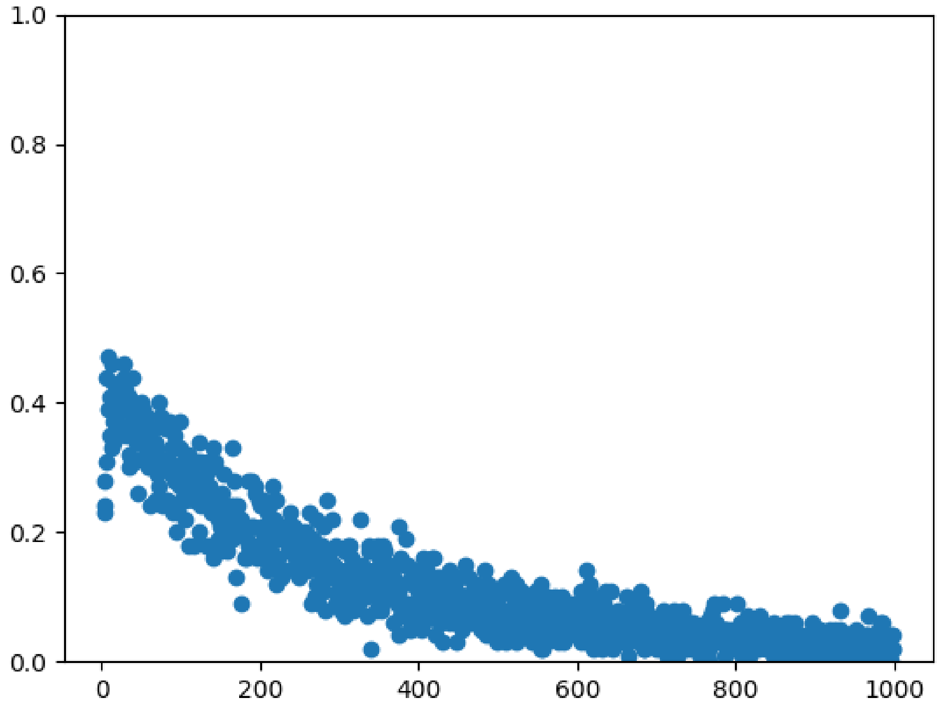
1(b).(ii).(1) : N = 20



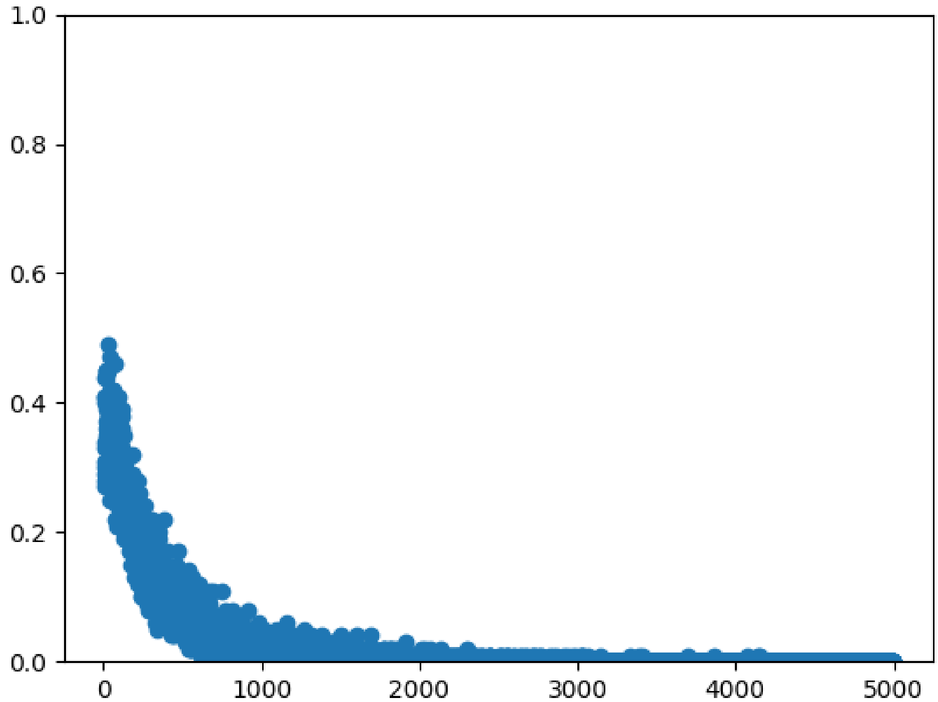
1(b).(i).(2) : N = 100



1(b).(ii).(3) : N = 1000



1(b).(i).(4) : N = 5000



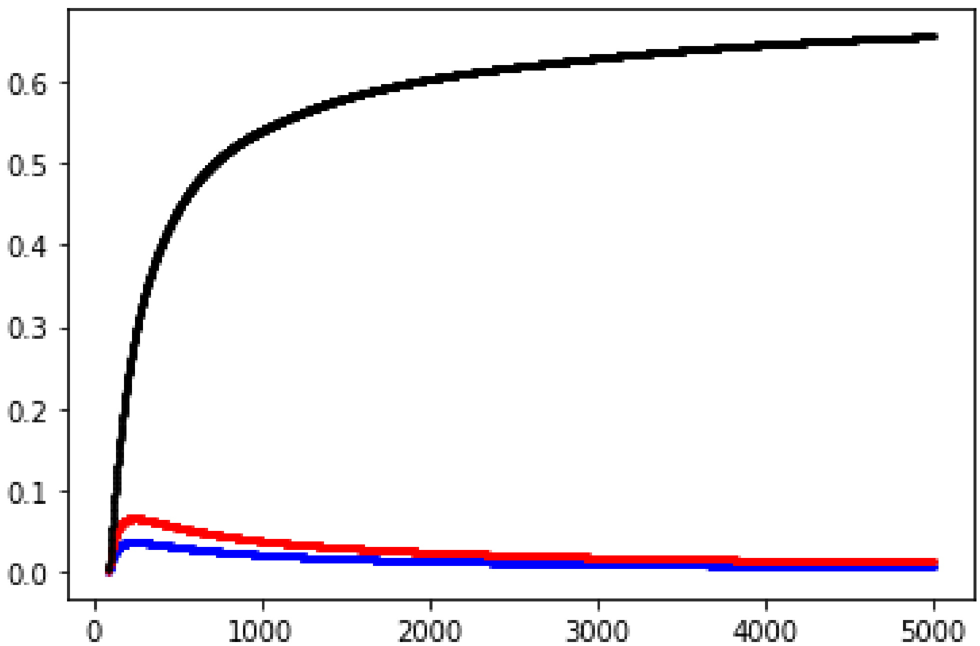
Problem 2

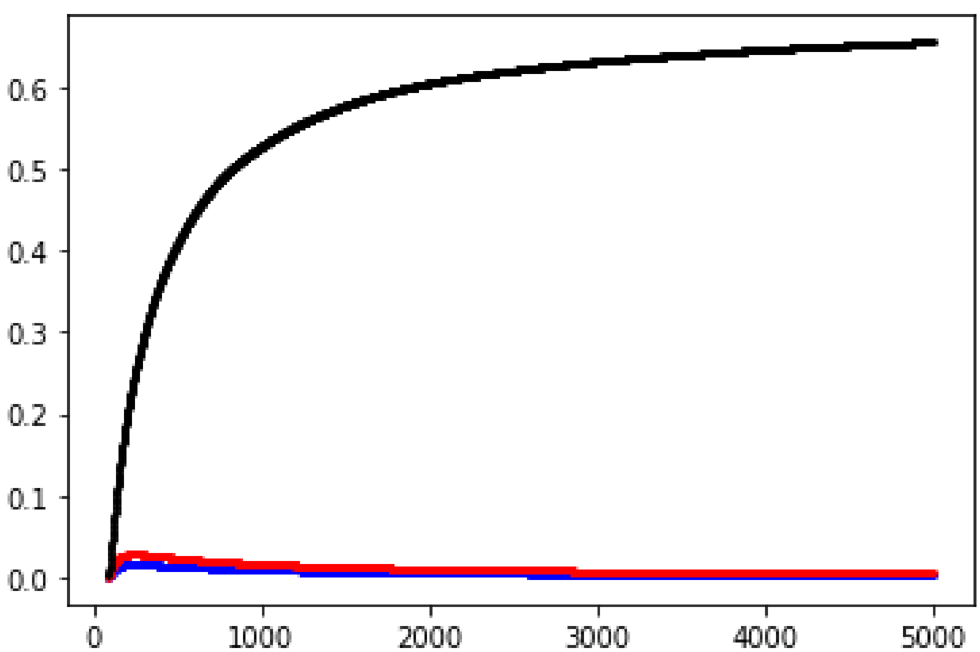
2(a).(i) :

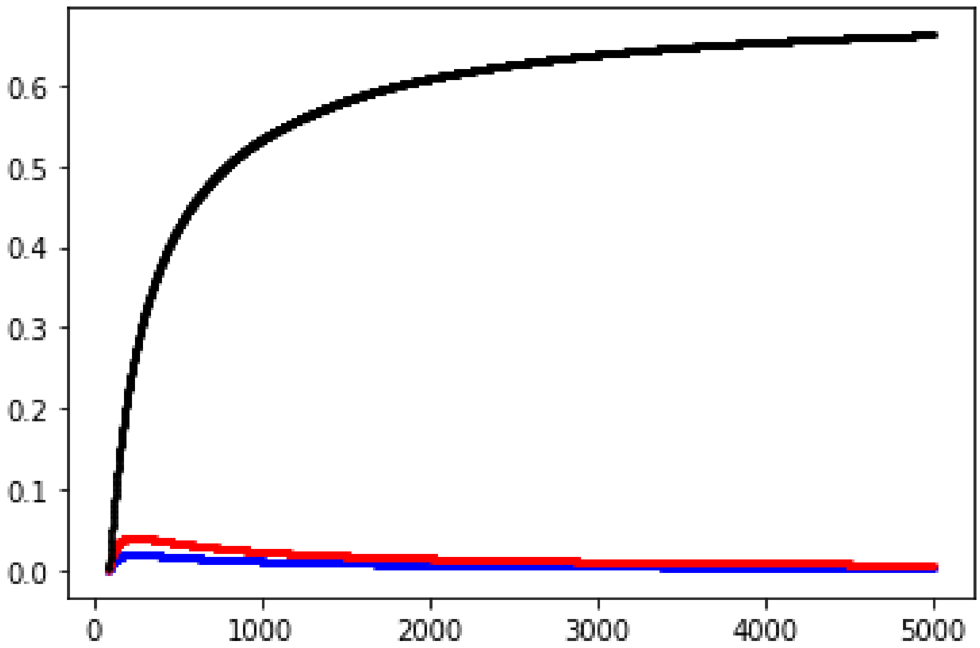
```
from cProfile import label
import math
from turtle import color
import matplotlib.pyplot as plt
from statistics import mean
import random
import numpy as np
for I in range(0,3):
    smk = [0, 0, 0]
    kA,kB,kC,nA,nB,nC,h, sma, smb, smc=0,0,0,0,0,0,0,0,0,0
    p_abc=[0.2,0.4,0.7]#pA,pB,pC
    alpha = [0.1,0.05,0.01]
    y = math.sqrt(math.log(1/alpha[I]))/math.sqrt(2)
    fig, ax = plt.subplots()
    smk_arra,smk_arrb,smk_arrc=[],[],[]
    for i in range(0,30):
        r = random.random()
        if (r>=p_abc[0]):
            kA+=0
            nA+=1
        else:
```

```python
            kA+=1
            nA+=1
        if (r>=p_abc[1]):
            kB+=0
            nB+=1
        else:
            kB+=1
            nB+=1
        if (r>=p_abc[2]):
            kC+=0
            nC+=1
        else:
            kC+=1
            nC+=1

    for i in range(90,5000):
        UCB_A = y/math.sqrt(nA)+kA/nA
        UCB_B = y/math.sqrt(nB)+kB/nB
        UCB_C = y/math.sqrt(nC)+kC/nC
        if (UCB_A>=UCB_B)and(UCB_A>=UCB_C):
            r = random.random()
            if (r>=p_abc[0]):
                kA+=0
                nA+=1
            else:
                kA+=1
                nA+=1
        elif (UCB_B>=UCB_A)and(UCB_B>=UCB_C):
            r = random.random()
            if (r>=p_abc[1]):
                kB+=0
                nB+=1
            else:
                kB+=1
                nB+=1
        elif (UCB_C>=UCB_B)and(UCB_C>=UCB_A):
            r = random.random()
            if (r>=p_abc[2]):
                kC+=0
                nC+=1
            else:
                kC+=1
                nC+=1
        sma += kA/i
        smb += kB/i
        smc += kC/i
        smk_arra.append(sma/i)
        smk_arrb.append(smb/i)
        smk_arrc.append(smc/i)
```

```
h = kA+kB+kC
x=np.linspace(90,5000,num=4910)
ax.scatter(x,smk_arra,c = "blue",s=2,vmax=0,vmin=180)
ax.scatter(x,smk_arrb,c = "red",s=2)
ax.scatter(x,smk_arrc,c = "black",s=2)
plt.show()
```



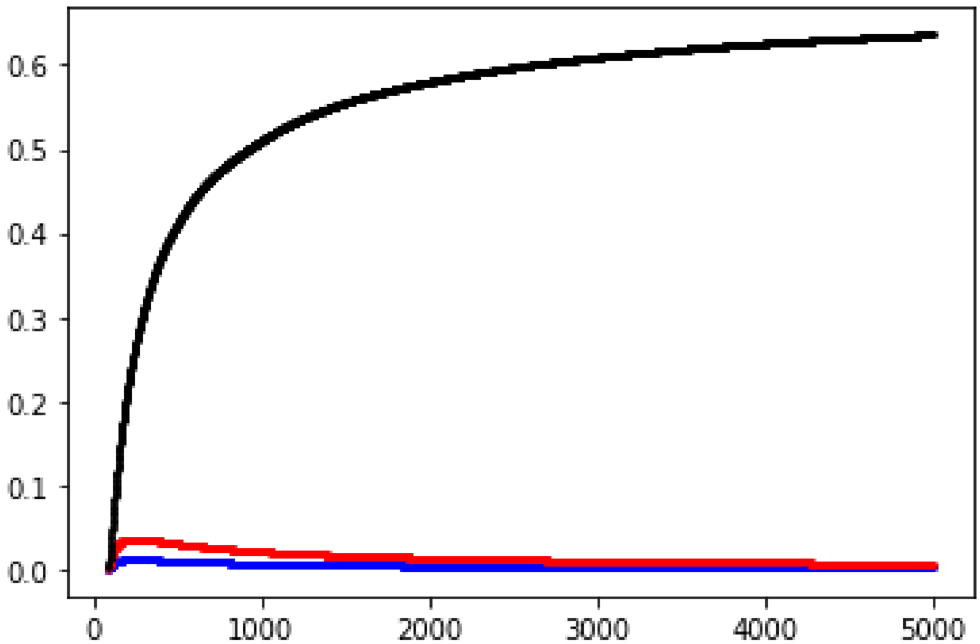




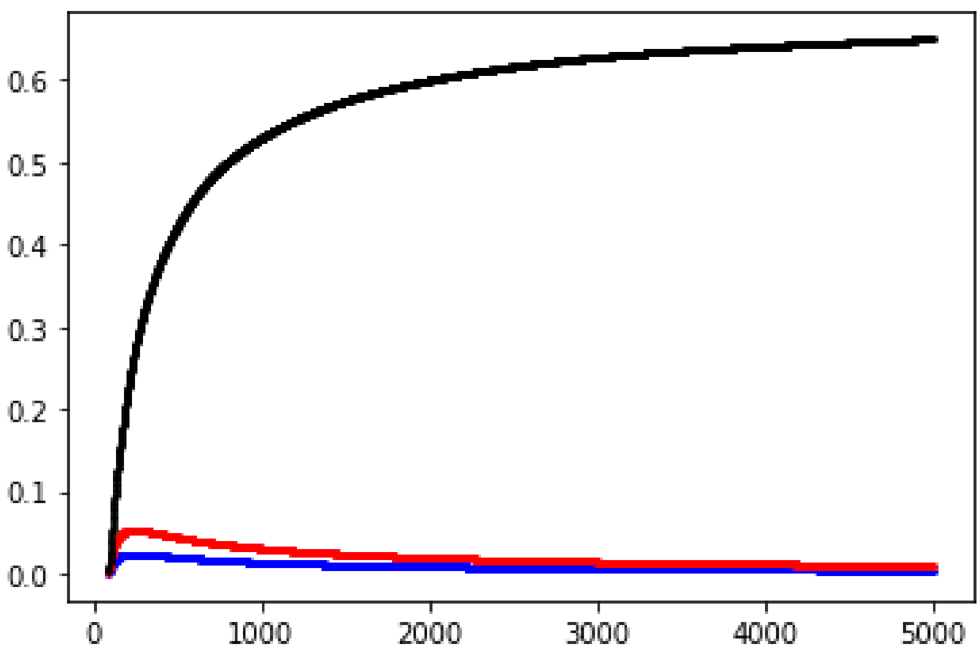
2(a).(ii) :

```
for I in range(0,3):
    smk = [0, 0, 0]
    kA,kB,kC,nA,nB,nC,h, sma, smb, smc=0,0,0,0,0,0,0,0,0,0
    p_abc=[0.2,0.4,0.7]#pA,pB,pC
    alpha = [0.1,0.05,0.01]
    y = math.sqrt(math.log(1/alpha[I]))/math.sqrt(2)
    fig, ax = plt.subplots()
    smk_arra,smk_arrb,smk_arrc=[],[],[]
    for i in range(0,30):
        r = random.random()
        if (r>=p_abc[0]):
            kA+=0
            nA+=1
        else:
            kA+=1
            nA+=1
        if (r>=p_abc[1]):
            kB+=0
            nB+=1
        else:
            kB+=1
            nB+=1
        if (r>=p_abc[2]):
            kC+=0
            nC+=1
        else:
            kC+=1
            nC+=1
```

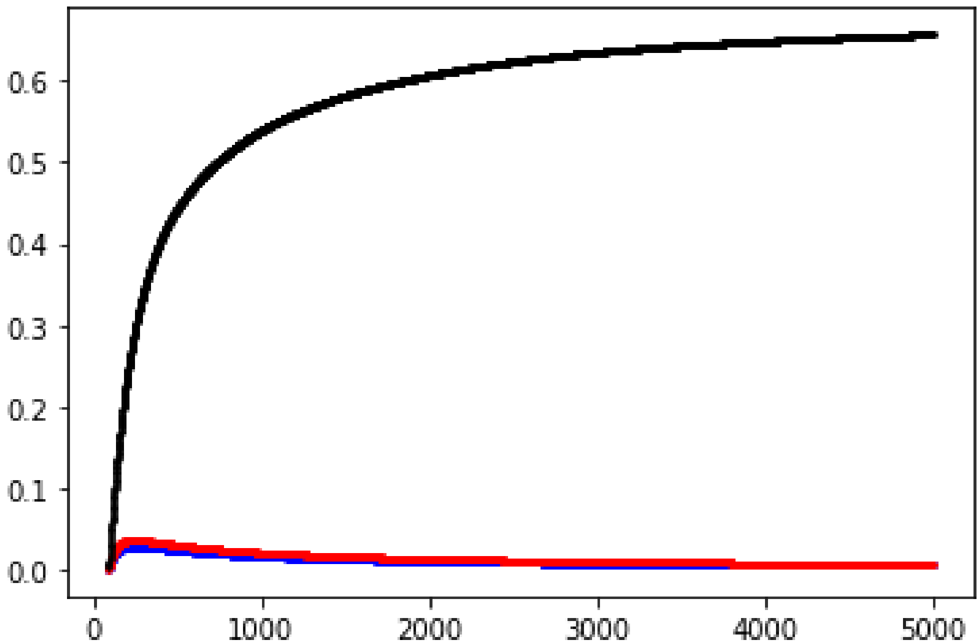
```python
for i in range(90,5000):
    UCB_A = y/math.sqrt(nA)+kA/nA
    UCB_B = y/math.sqrt(nB)+kB/nB
    UCB_C = y/math.sqrt(nC)+kC/nC
    if (UCB_A>=UCB_B)and(UCB_A>=UCB_C):
        r = random.random()
        if (r>=p_abc[0]):
            kA+=0
            nA+=1
        else:
            kA+=1
            nA+=1
    elif (UCB_B>=UCB_A)and(UCB_B>=UCB_C):
        r = random.random()
        if (r>=p_abc[1]):
            kB+=0
            nB+=1
        else:
            kB+=1
            nB+=1
    elif (UCB_C>=UCB_B)and(UCB_C>=UCB_A):
        r = random.random()
        if (r>=p_abc[2]):
            kC+=0
            nC+=1
        else:
            kC+=1
            nC+=1
    sma += kA/i
    smb += kB/i
    smc += kC/i
    smk_arra.append(sma/i)
    smk_arrb.append(smb/i)
    smk_arrc.append(smc/i)

h = kA+kB+kC
x=np.linspace(90,5000,num=4910)
ax.scatter(x,smk_arra,c = "blue",s=2,vmax=0,vmin=180)
ax.scatter(x,smk_arrb,c = "red",s=2)
ax.scatter(x,smk_arrc,c = "black",s=2)
plt.show()
```



0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1000
2000
3000
4000
5000



0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1000
2000
3000
4000
5000



2(b):

```
from cProfile import label
import math
from turtle import color
# import matplotlib.pyplot as plt
from statistics import mean
import random
import numpy as np
print(" "*(3),"N"," ","alpha"," ","pA,pB,pC"," "," "*2*(4-
len(str(p_abc[0]))),"  "*(4),"Sample average", " ", "Best expected
value")
print()
alpha = [0.1,0.05,0.01]
PABC=[[0.2,0.4,0.7],[0.45,0.5,0.58]]
nm=[(20,6),(100,18),(1000,60),(5000,90)]
for (N,m) in nm:
  for p_abc in PABC:
    for alp in alpha:
        kA,kB,kC,nA,nB,nC,h=0,0,0,0,0,0,0
        y = math.sqrt(math.log(1/alp))/math.sqrt(2)
        # fig, ax = plt.subplots()
        k_arra,k_arrb,k_arrc=[],[],[]
        for i in range(0,m//3):
            r = random.random()
            if (r>=p_abc[0]):
                kA+=0
                nA+=1
            else:
                kA+=1
```

```python
            nA+=1
        if (r>=p_abc[1]):
            kB+=0
            nB+=1
        else:
            kB+=1
            nB+=1
        if (r>=p_abc[2]):
            kC+=0
            nC+=1
        else:
            kC+=1
            nC+=1

    for i in range(m,N):
        UCB_A = y/math.sqrt(nA)+kA/nA
        UCB_B = y/math.sqrt(nB)+kB/nB
        UCB_C = y/math.sqrt(nC)+kC/nC
        if (UCB_A>=UCB_B)and(UCB_A>=UCB_C):
            r = random.random()
            if (r>=p_abc[0]):
                kA+=0
                nA+=1
            else:
                kA+=1
                nA+=1
        elif (UCB_B>=UCB_A)and(UCB_B>=UCB_C):
            r = random.random()
            if (r>=p_abc[1]):
                kB+=0
                nB+=1
            else:
                kB+=1
                nB+=1
        elif (UCB_C>=UCB_B)and(UCB_C>=UCB_A):
            r = random.random()
            if (r>=p_abc[2]):
                kC+=0
                nC+=1
            else:
                kC+=1
                nC+=1
        k_arra.append(kA/i)
        k_arrb.append(kB/i)
        k_arrc.append(kC/i)

    h = kA+kB+kC
    # x=np.linspace(90,5000,num=4910)
    # ax.scatter(x,k_arra,c = "blue",s=2,vmax=0,vmin=180)
    # ax.scatter(x,k_arrb,c = "red",s=2)
```

```
        # ax.scatter(x,k_arrc,c = "black",s=2)
        # plt.show()

        print(" "*(4-len(str(N))),N," "," "*(4-len(str(alp))),alp,"
",p_abc," "," "*2*(4-len(str(p_abc[0])))," "*(6-len(str(h/N))),h/N, "
"*4, p_abc[2])
```

```
    N    alpha    pA,pB,pC         Sample average   Best expected value

   20     0.1    [0.2, 0.4, 0.7]          0.55      0.7
   20    0.05    [0.2, 0.4, 0.7]          0.75      0.7
   20    0.01    [0.2, 0.4, 0.7]           0.5      0.7
   20     0.1    [0.45, 0.5, 0.58]        0.75      0.58
   20    0.05    [0.45, 0.5, 0.58]         0.4      0.58
   20    0.01    [0.45, 0.5, 0.58]        0.75      0.58
  100     0.1    [0.2, 0.4, 0.7]          0.66      0.7
  100    0.05    [0.2, 0.4, 0.7]          0.71      0.7
  100    0.01    [0.2, 0.4, 0.7]          0.67      0.7
  100     0.1    [0.45, 0.5, 0.58]        0.66      0.58
  100    0.05    [0.45, 0.5, 0.58]        0.52      0.58
  100    0.01    [0.45, 0.5, 0.58]        0.46      0.58
 1000     0.1    [0.2, 0.4, 0.7]          0.66      0.7
 1000    0.05    [0.2, 0.4, 0.7]         0.684      0.7
 1000    0.01    [0.2, 0.4, 0.7]         0.668      0.7
 1000     0.1    [0.45, 0.5, 0.58]       0.589      0.58
 1000    0.05    [0.45, 0.5, 0.58]       0.529      0.58
 1000    0.01    [0.45, 0.5, 0.58]       0.563      0.58
 5000     0.1    [0.2, 0.4, 0.7]        0.6922      0.7
 5000    0.05    [0.2, 0.4, 0.7]        0.6966      0.7
 5000    0.01    [0.2, 0.4, 0.7]        0.6912      0.7
 5000     0.1    [0.45, 0.5, 0.58]      0.5586      0.58
 5000    0.05    [0.45, 0.5, 0.58]      0.5758      0.58
 5000    0.01    [0.45, 0.5, 0.58]      0.5766      0.58
```

2(c) :

Increasing N brings the sample average closer to best expected value. Decreasing α in general increases the sample average. When p_A, p_B, and p_C are very close, then the sample average diverges more from best expected value and vice-versa.