INF 559: Homework 1

Shaoqian Chen

8831737894

```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

```
import numpy as np
import matplotlib.pyplot as plt
import time
import random
np.set_printoptions(suppress=True) # Suppress scientific notation while printing numb
%matplotlib inline
```

2. Plotting latency and bandwidth

a. Open the test file in unbuffered mode

```
testdata = open("gdrive/My Drive/ubuntu-18.04.3-desktop-amd64.iso",mode="rb",bufferin
```

```
testdata
```

```
<_io.FileIO name='gdrive/My Drive/ubuntu-18.04.3-desktop-amd64.iso' mode='rb' clo
```

b. Sequentially read [1, 4, 16, 64, 256, 1024, 4*1024, 16*1024, 64*1024, 256*1024, 1024*1024] size of 4KB. Measure the latency for each iteration in terms of wall-clock time.

```
Q2 = testdata
blocks = [1,4,16,64,256,1024,4*1024,16*1024,64*1024,256*1024,1024*1024]
size = 4
len(blocks)
```

```
11
```

```
wall_clock_seq = []
for i in range(len(blocks)):
    start = time.time()
```

```
  start = time.time()
  Q2.readline(blocks[i]*4)
  end = time.time()
  wall_clock_seq.append(end-start)
```

```
wall_clock_seq
```

```
[0.00079345703125,
 1.6927719116210938e-05,
 4.8160552978515625e-05,
 0.00013756752014160156,
 6.723403930664062e-05,
 1.049041748046875e-05,
 0.012387275695800781,
 0.014492034912109375,
 0.00021076202392578125,
 0.0008184909820556641,
 0.0014600753784179688]
```

c. Repeat 2b with random reads instead of sequential.

```
rand_pos = random.sample(range(0,11), 11)
```

```
rand_pos
```

```
[9, 6, 10, 0, 5, 3, 1, 2, 4, 7, 8]
```

```
wall_clock_rand = []
for j in rand_pos:
  begin = time.time()
  Q2.readline(blocks[j]*4)
  over = time.time()
  wall_clock_rand.append(over-begin)
```

```
wall_clock_rand
```

```
[0.007957696914672852,
 0.0001499652862548828,
 0.0002536773681640625,
 7.867813110351562e-06,
 0.0020613670349121094,
 0.00024175643920898438,
 2.9802322387695312e-05,
 9.274482727050781e-05,
 0.0002567768096923828,
 0.0009310245513916016,
 0.000270843505859375]
```

```
wall_clock_seq
```

```
[0.00079345703125,
 1.6927719116210938e-05,
 4.8160552978515625e-05,
 0.00013756752014160156,
 6.723403930664062e-05,
 1.049041748046875e-05,
 0.012387275695800781,
 0.014492034912109375,
 0.00021076202392578125,
 0.0008184909820556641,
 0.0014600753784179688]
```

d. Plot the latencies measured in 2b and 2c against the number of blocks read. B
should appear on the same plot and the number of blocks should be scaled logar
Briefly describe your observations from this plot.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd



lat_seq = pd.DataFrame(blocks)
lat_seq.rename(columns={0:'No. of blocks'})
lat_seq['Latency(seconds)'] = wall_clock_seq
Lat_Seq = lat_seq.rename(columns={0:'No.of blocks'})
Lat_Seq
```

	No.of blocks	Latency(seconds)
0	1	0.000793
1	4	0.000017
2	16	0.000048
3	64	0.000138
4	256	0.000067
5	1024	0.000010
6	4096	0.012387
7	16384	0.014492
8	65536	0.000211
9	262144	0.000818
10	1048576	0.001460

```
#convert random latency data into dataframe
rand_pos
rand_pos_4 = []
for m in rand_pos:
  rand_pos_4.append(4**m)
rand_pos_4
lat_rand = pd.DataFrame(rand_pos_4)
latency = lat_rand.rename(columns={0:'No. of blocks'})
latency['Latency(seconds)_Random'] = wall_clock_rand
LATENCY = latency.sort_values(by=['No. of blocks'])
LATENCY
```

	No. of blocks	Latency(seconds)_Random
3	1	0.000008
6	4	0.000030
7	16	0.000093
5	64	0.000242
8	256	0.000257
4	1024	0.002061
1	4096	0.000150
9	16384	0.000931
10	65536	0.000271
0	262144	0.007958
2	1048576	0.000254

```
LATENCY['Latency(seconds)_Sequential'] = wall_clock_seq
```

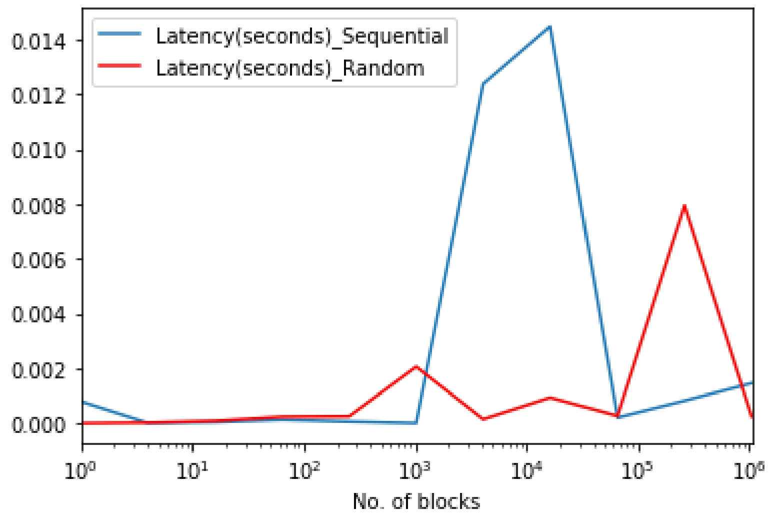
```
LATENCY
```

	No. of blocks	Latency(seconds)_Random	Latency(seconds)_Sequential
3	1	0.000008	0.000793
6	4	0.000030	0.000017
7	16	0.000093	0.000048
5	64	0.000242	0.000138
8	256	0.000257	0.000067
4	1024	0.002061	0.000010
1	4096	0.000150	0.012387
9	16384	0.000931	0.014492
10	65536	0.000271	0.000211
0	262144	0.007958	0.000818
2	1048576	0.000254	0.001460

```
test = LATENCY
test['Blocks'] = ['4^0','4^1','4^2','4^3','4^4','4^5','4^6','4^7','4^8','4^9','4^10',
test
```

	No. of blocks	Latency(seconds)_Random	Latency(seconds)_Sequential	Blocks
3	1	0.000008	0.000793	4^0
6	4	0.000030	0.000017	4^1
7	16	0.000093	0.000048	4^2
5	64	0.000242	0.000138	4^3
8	256	0.000257	0.000067	4^4
4	1024	0.002061	0.000010	4^5
1	4096	0.000150	0.012387	4^6
9	16384	0.000931	0.014492	4^7
10	65536	0.000271	0.000211	4^8
0	262144	0.007958	0.000818	4^9
2	1048576	0.000254	0.001460	4^10

```
ax = plt.gca()
test.plot(kind='line',x='No. of blocks',y='Latency(seconds)_Sequential',ax=ax)
test.plot(kind='line',x='No. of blocks',y='Latency(seconds)_Random', color='red',ax=a
ax.set_xscale('log')
plt.show()
```



e. Calculate the bandwidth for each iteration of 2b and 2c using the latency and a
the results in the same manner as latency and briefly describe your observations.

```
#Bandwidth is in byte/second
```

```
bandwidth_df = test
bandwidth_df
```

	No. of blocks	Latency(seconds)_Random	Latency(seconds)_Sequential	Blocks
3	1	0.000008	0.000793	4^0
6	4	0.000030	0.000017	4^1
7	16	0.000093	0.000048	4^2
5	64	0.000242	0.000138	4^3
8	256	0.000257	0.000067	4^4
4	1024	0.002061	0.000010	4^5
1	4096	0.000150	0.012387	4^6
9	16384	0.000931	0.014492	4^7
10	65536	0.000271	0.000211	4^8
0	262144	0.007958	0.000818	4^9
2	1048576	0.000254	0.001460	4^10

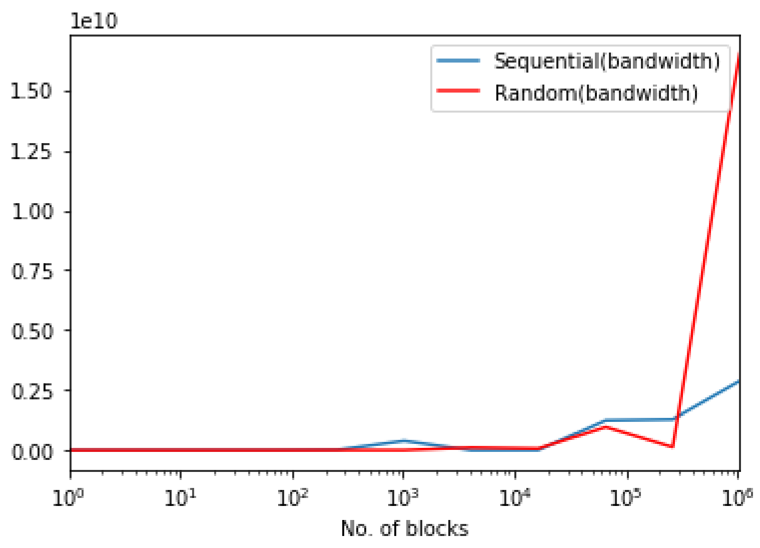
```
bandwidth_df['Sequential(bandwidth)'] = bandwidth_df['No. of blocks']*4/bandwidth_df[
```

```
bandwidth_df['Random(bandwidth)'] = bandwidth_df['No. of blocks']*4/bandwidth_df['Lat
```

```
bandwidth_df
```

	No. of blocks	Latency(seconds)_Random	Latency(seconds)_Sequential	Blocks
3	1	0.000008	0.000793	4^0
6	4	0.000030	0.000017	4^1
7	16	0.000093	0.000048	4^2
5	64	0.000242	0.000138	4^3
8	256	0.000257	0.000067	4^4
4	1024	0.002061	0.000010	4^5
1	4096	0.000150	0.012387	4^6
9	16384	0.000931	0.014492	4^7
10	65536	0.000271	0.000211	4^8
0	262144	0.007958	0.000818	4^9
2	1048576	0.000254	0.001460	4^10

```
ax = plt.gca()
ax = plt.gca()
bandwidth_df.plot(kind='line',x='No. of blocks',y='Sequential(bandwidth)',ax=ax)
bandwidth_df.plot(kind='line',x='No. of blocks',y='Random(bandwidth)', color='red',ax
ax.set_xscale('log')
plt.show()
```



3. Monte-Carlo simulations

a. Run 10 simulations of steps 2b and 2c and store the results.

```
#2b
"""
For each of the dataframe created below, each column represent a simulation, each
row represents a single iteration within that simulation.

              simulation1   simulation2   simulation3...   simulation10
iteration 1
iteration 2
.
.
.
iteration 10
"""
sim_seq = pd.DataFrame()# empty dataframe store all simulation of Sequential case
sim_rand = pd.DataFrame()# empty dataframe store all simulation of Random case
for sim in range(10):
  temp_seq = []
  temp_rand = []
  rand_pos = random.sample(range(0,11), 11)
  for i in range(len(blocks)):
    #add iteration into a temp Squential list
    start = time.time()
    Q2.readline(blocks[i]*4)
    end = time.time()
    temp_seq.append(end-start)
    #add iteration into a temp Random list
    begin = time.time()
    Q2.readline(blocks[rand_pos[i]]*4)
    over = time.time()
    temp_rand.append(over-begin)
    [temp_rand for _,temp_rand in sorted(zip(rand_pos,temp_rand))]

  sim_seq[sim] = temp_seq
  sim_rand[sim] = temp_rand
sim_rand
```

	0	1	2	3	4	5	6	7	8
0	0.000009	0.000004	0.000003	0.000523	0.000044	0.000019	0.000088	0.000053	0.000012
1	0.000375	0.000262	0.000071	0.000211	0.000040	0.001903	0.000037	0.000013	0.000077
2	0.000011	0.000155	0.000717	0.000984	0.000005	0.000148	0.000097	0.000002	0.000037
3	0.000136	0.000066	0.000375	0.000150	0.000034	0.005462	0.000037	0.000001	0.000089
4	0.000654	0.000006	0.000367	0.000003	0.000124	0.000037	0.000029	0.000016	0.000005
5	0.000005	0.000680	0.000149	0.000083	0.000003	0.000002	0.000036	0.000010	0.000002
6	0.000005	0.000088	0.000418	0.000190	0.000006	0.000002	0.000003	0.000001	0.000024
7	0.000093	0.000366	0.000242	0.001774	0.000599	0.000001	0.000016	0.000002	0.000059
8	0.000411	0.000136	0.000766	0.000038	0.000038	0.000017	0.000037	0.000002	0.000022
9	0.000088	0.000360	0.000110	0.000010	0.002491	0.000036	0.000010	0.000008	0.000003
10	0.000166	0.000070	0.000014	0.000051	0.005741	0.000013	0.000016	0.000001	0.000004

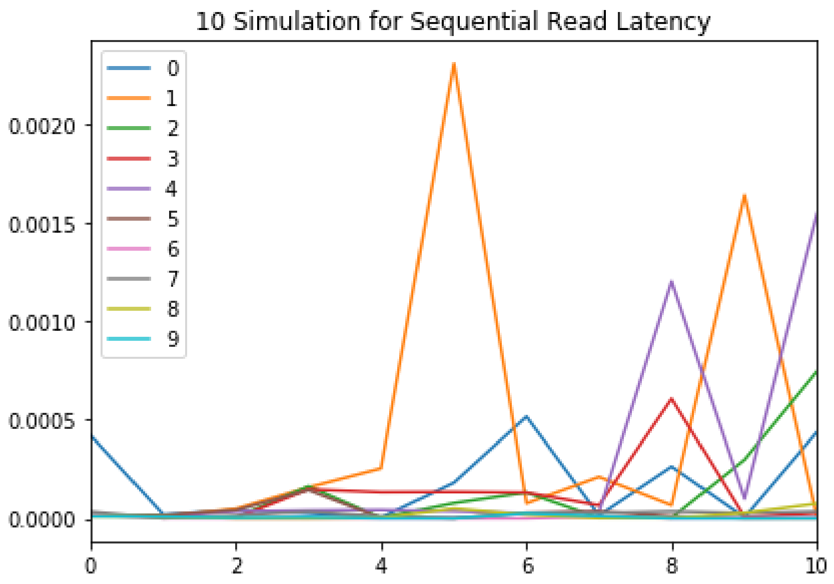
```
sim_seq
```

	0	1	2	3	4	5	6	7	8
0	0.000422	0.000019	0.000017	0.000018	0.000016	0.000020	0.000020	0.000036	0.00001
1	0.000021	0.000012	0.000012	0.000012	0.000010	0.000018	0.000002	0.000002	0.00001
2	0.000044	0.000051	0.000006	0.000005	0.000037	0.000044	0.000001	0.000016	0.00000
3	0.000027	0.000159	0.000165	0.000149	0.000042	0.000145	0.000002	0.000036	0.00000
4	0.000006	0.000255	0.000005	0.000134	0.000044	0.000004	0.000001	0.000013	0.00000
5	0.000183	0.002307	0.000079	0.000136	0.000038	0.000003	0.000001	0.000001	0.00005
6	0.000517	0.000078	0.000133	0.000133	0.000027	0.000030	0.000002	0.000028	0.00002
7	0.000016	0.000212	0.000005	0.000068	0.000033	0.000037	0.000011	0.000029	0.00000
8	0.000264	0.000069	0.000005	0.000607	0.001203	0.000013	0.000001	0.000037	0.00000
9	0.000006	0.001641	0.000299	0.000009	0.000101	0.000002	0.000002	0.000029	0.00002
10	0.000440	0.000005	0.000747	0.000028	0.001549	0.000001	0.000004	0.000036	0.00007

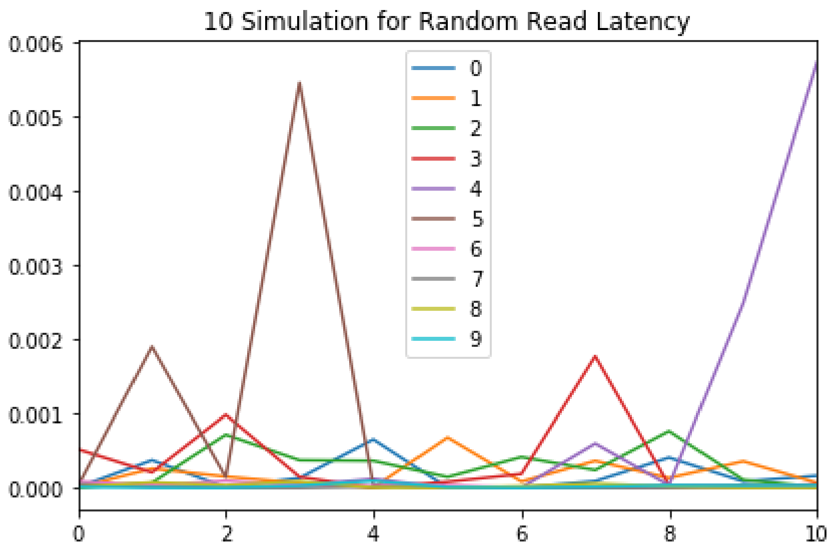
```
# Issue Encountered:
"""
* Each iteration of the random process is using different sequence, need to be
  specific saperately
* Sequential case works great
"""
#PROBLEM SOLVED
```

```
'\n* Each iteration of the random process is using different sequence, need to be
```

```
sim_seq.plot(subplots=False)
plt.tight_layout()
plt.title("10 Simulation for Sequential Read Latency")
plt.show()
```



```
sim_rand.plot(subplots=False)
plt.title("10 Simulation for Random Read Latency")
plt.tight_layout()
plt.show()
```



```
"""
Sort list b according to list a
"""
a = [3,1,2,4,0,12]
b = [0.3,0.1,0.2,0.4,0,1.2]
[b for _,b in sorted(zip(a,b))]
```

```
[0, 0.1, 0.2, 0.3, 0.4, 1.2]
```

b. For each of the 11 iterations, calculate the mean and standard error4 over the

```
sim_rand_mean = sim_rand.mean(1)
sim_rand_se = sim_rand.sem(1)
sim_rand_data = pd.DataFrame()
sim_rand_data['Mean'] = sim_rand_mean
sim_rand_data['Standard Error'] = sim_rand_se
```

```
sim_seq_mean = sim_seq.mean(1)
sim_seq_se = sim_seq.sem(1)
sim_seq_data = pd.DataFrame()
sim_seq_data['Mean'] = sim_seq_mean
sim_seq_data['Standard Error'] = sim_seq_se
sim_rand_data
```

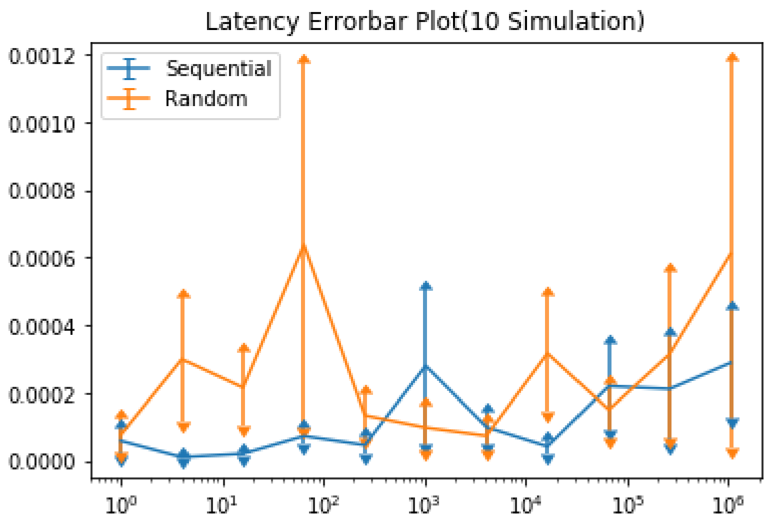
	Mean	Standard Error
0	0.000077	0.000050
1	0.000299	0.000182
2	0.000216	0.000109
3	0.000638	0.000537
4	0.000134	0.000068
5	0.000098	0.000066
6	0.000074	0.000043
7	0.000317	0.000173
8	0.000150	0.000078
9	0.000315	0.000244
10	0.000611	0.000570

```
sim_seq_data
```

	Mean	Standard Error
0	0.000059	0.000040
1	0.000011	0.000002
2	0.000021	0.000006
3	0.000073	0.000023
4	0.000047	0.000027
5	0.000280	0.000226
6	0.000099	0.000049
7	0.000043	0.000020
8	0.000220	0.000125
9	0.000212	0.000161
10	0.000289	0.000161

c. Use the results of 3b to generate errorbar plots for latency and bandwidth. Aga
results should be on the same plot and the number of blocks should be scaled lo
your observations from these plots.

```
ax = plt.gca()
plt.errorbar(test['No. of blocks'],sim_seq_data.Mean,yerr=sim_seq_data['Standard Erro
plt.errorbar(test['No. of blocks'],sim_rand_data.Mean,yerr=sim_rand_data['Standard Er
ax.set_xscale('log')
plt.title('Latency Errorbar Plot(10 Simulation)')
plt.legend()
plt.show()
```



```
bandwidth_rand = pd.DataFrame()
for i in range(10):
  bandwidth_rand[i] = test['No. of blocks']*4/sim_rand[i]
```

```
bandwidth_rand
```

	0	1	2	3	4	5	
0	1.188661e+11	2.443359e+11	3.998224e+11	2.003666e+09	2.364541e+10	5.567147e+10	1.1
1	4.368689e+07	6.241551e+07	2.298310e+08	7.773696e+07	4.090445e+08	8.610384e+06	4.3
2	3.665039e+11	2.710660e+10	5.846522e+09	4.262706e+09	8.796093e+11	2.837449e+10	4.3
3	2.948544e+04	6.056757e+04	1.066574e+04	2.667284e+04	1.165084e+05	7.323737e+02	1.0
4	6.258604e+06	6.362915e+08	1.114852e+07	1.227134e+09	3.316577e+07	1.094259e+08	1.4
5	5.113056e+07	3.764873e+05	1.715243e+06	3.094357e+06	8.259552e+07	1.342177e+08	7.0
6	3.355443e+06	1.823610e+05	3.826047e+04	8.399107e+04	2.581110e+06	9.586981e+06	4.7
7	6.882960e+05	1.746490e+05	2.644684e+05	3.608488e+04	1.067762e+05	5.368709e+07	3.8
8	2.494174e+06	7.521834e+06	1.336747e+06	2.701237e+07	2.684355e+07	6.049250e+07	2.7
9	7.409108e+08	1.819179e+08	5.936888e+08	6.544712e+09	2.630914e+07	1.832519e+09	6.5
10	1.582031e+09	3.739835e+09	1.928968e+10	5.162026e+09	4.566457e+07	2.036133e+10	1.6

```
bandwidth_seq = pd.DataFrame()
for j in range(10):
  bandwidth_seq[j] = test['No. of blocks']*4/sim_seq[j]
bandwidth_seq
```

	0	1	2	3	4	5	
0	2.481968e+09	5.638521e+10	6.194432e+10	5.864062e+10	6.467715e+10	5.298851e+10	5.2
1	7.898790e+08	1.402438e+09	1.321528e+09	1.347441e+09	1.561806e+09	9.162597e+08	7.6
2	9.509290e+10	8.259242e+10	6.515624e+11	8.377231e+11	1.127704e+11	9.613216e+10	2.9
3	1.458888e+05	2.511559e+04	2.431481e+04	2.688656e+04	9.478653e+04	2.750363e+04	2.3
4	7.158279e+08	1.607097e+07	8.589935e+08	3.046076e+07	9.286416e+07	1.010581e+09	2.8
5	1.399924e+06	1.109581e+05	3.224450e+06	1.887068e+06	6.795834e+06	9.761289e+07	2.1
6	3.096856e+04	2.046002e+05	1.200516e+05	1.206994e+05	5.835553e+05	5.284163e+05	9.5
7	3.890369e+06	3.019521e+05	1.412818e+07	9.353152e+05	1.917396e+06	1.731842e+06	5.9
8	3.883334e+06	1.491308e+07	2.147484e+08	1.686285e+06	8.511628e+05	7.953643e+07	7.1
9	1.018066e+10	3.992997e+07	2.193758e+08	7.429133e+09	6.467715e+08	3.926827e+10	2.7
10	5.962644e+08	4.997780e+10	3.509453e+08	9.397535e+09	1.692859e+08	1.832519e+11	6.4

```
bandwidth_rand_mean = bandwidth_rand.mean(1)
bandwidth_rand_se = bandwidth_rand.sem(1)
bandwidth_rand_data = pd.DataFrame()
bandwidth_rand_data['Mean'] = bandwidth_rand_mean
bandwidth_rand_data['Standard Error'] = bandwidth_rand_se
```

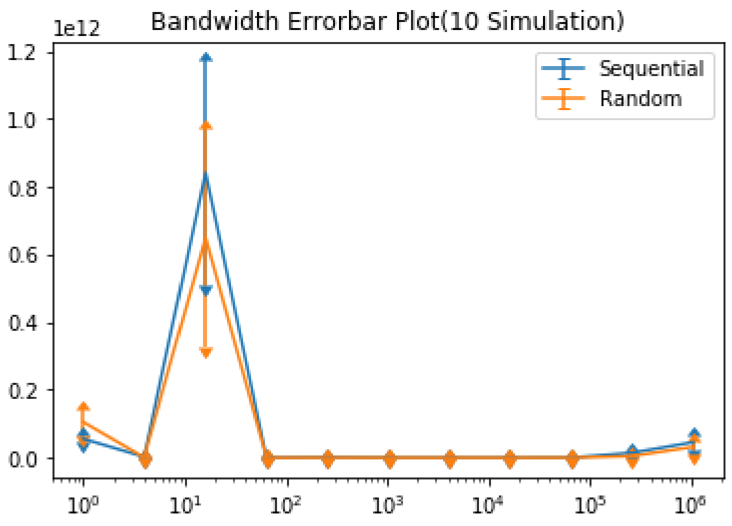
```
bandwidth_seq_mean = bandwidth_seq.mean(1)
bandwidth_seq_se = bandwidth_seq.sem(1)
bandwidth_seq_data = pd.DataFrame()
bandwidth_seq_data['Mean'] = bandwidth_seq_mean
bandwidth_seq_data['Standard Error'] = bandwidth_seq_se
bandwidth_rand_data
```

	Mean	Standard Error
0	1.053671e+11	3.981219e+10
1	7.640423e+08	4.747678e+08
2	6.494658e+11	3.221172e+11
3	3.319265e+05	2.741738e+05
4	3.203019e+08	1.307934e+08
5	4.512527e+07	1.588971e+07
6	4.588615e+06	1.591553e+06
7	1.022802e+07	6.091596e+06
8	8.405579e+07	5.912490e+07
9	4.796002e+09	2.056154e+09
10	3.177794e+10	1.775003e+10

```
bandwidth_seq_data
```

	Mean	Standard Error
0	5.571436e+10	8.077423e+09
1	2.577028e+09	8.473206e+08
2	8.413756e+11	3.293869e+11
3	6.716192e+05	3.770713e+05
4	9.593658e+08	3.146505e+08
5	6.630763e+07	2.734945e+07
6	1.314165e+06	9.228664e+05
7	5.842941e+06	2.227603e+06
8	2.030637e+08	8.551019e+07
9	1.447540e+10	6.140918e+09
10	4.564976e+10	2.067349e+10

```
ax = plt.gca()
plt.errorbar(test['No. of blocks'],bandwidth_seq_data.Mean,yerr=bandwidth_seq_data['S
plt.errorbar(test['No. of blocks'],bandwidth_rand_data.Mean,yerr=bandwidth_rand_data[
ax.set_xscale('log')
plt.title('Bandwidth Errorbar Plot(10 Simulation)')
plt.legend()
plt.show()
```


Bandwidth Errorbar Plot(10 Simulation)
1e12
1.2
1.0
0.8
0.6
0.4
0.2
0.0
10⁰
10¹
10²
10³
10⁴
10⁵
10⁶
Sequential
Random