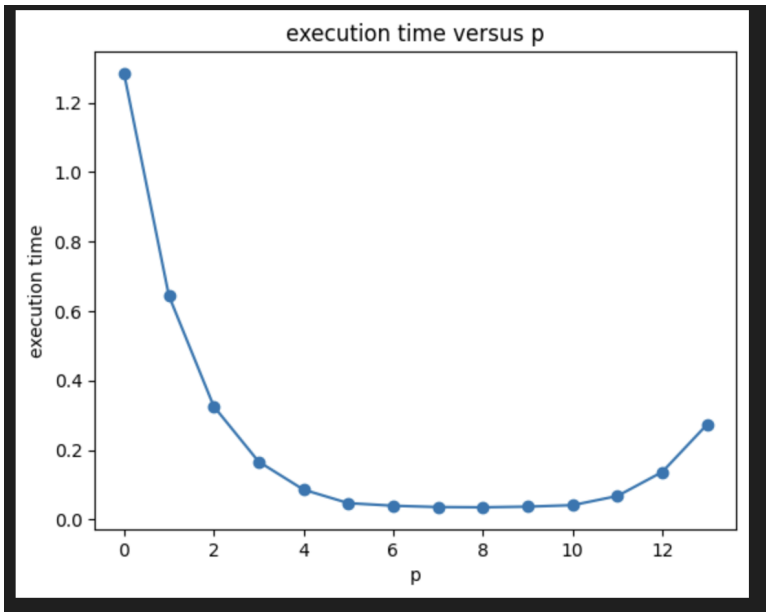
Chonglin Zhang
CSCE-735
DUE: 9-11-2023
833003072

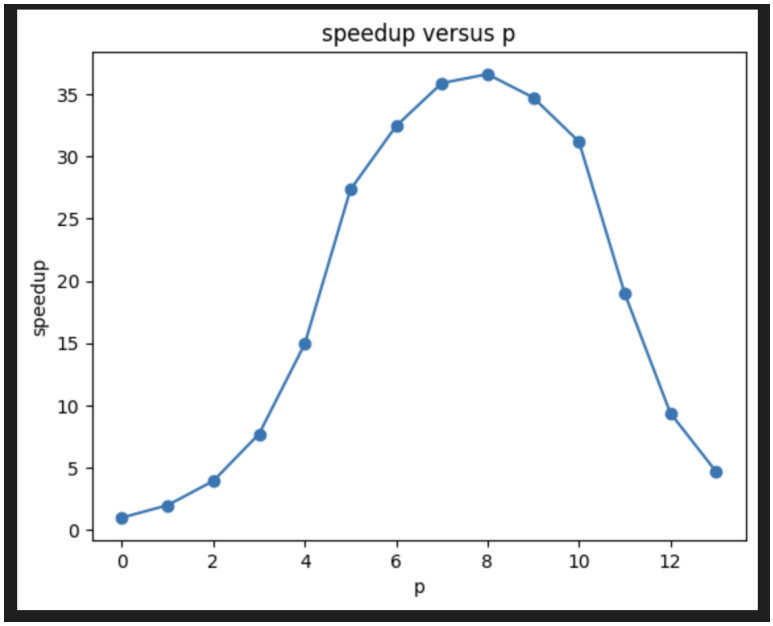
1. Execute the code for n=108 with p chosen to be 2^k, for k = 0, 1, …, 13. Using the experimental data obtained from these experiments, answer the following questions. For plots, use a logarithmic scale for the x-axis.

```
Trials = 100000000, Threads =    1, pi =   3.1416149600, error = 7.10e-06, time (sec) =   1.2843
Trials = 100000000, Threads =    2, pi =   3.1417022800, error = 3.49e-05, time (sec) =   0.6440
Trials = 100000000, Threads =    4, pi =   3.1415910800, error = 5.01e-07, time (sec) =   0.3258
Trials = 100000000, Threads =    8, pi =   3.1415947600, error = 6.70e-07, time (sec) =   0.1666
Trials = 100000000, Threads =   16, pi =   3.1415291200, error = 2.02e-05, time (sec) =   0.0860
Trials = 100000000, Threads =   32, pi =   3.1415901200, error = 8.06e-07, time (sec) =   0.0470
Trials = 100000000, Threads =   64, pi =   3.1413512400, error = 7.68e-05, time (sec) =   0.0396
Trials = 100000000, Threads =  128, pi =   3.1429299200, error = 4.26e-04, time (sec) =   0.0358
Trials = 100000000, Threads =  256, pi =   3.1412995200, error = 9.33e-05, time (sec) =   0.0351
Trials = 100000000, Threads =  512, pi =   3.1468450800, error = 1.67e-03, time (sec) =   0.0370
Trials = 100000000, Threads = 1024, pi =   3.1514026000, error = 3.12e-03, time (sec) =   0.0412
Trials = 100000000, Threads = 2048, pi =   3.1453611600, error = 1.20e-03, time (sec) =   0.0676
Trials = 100000000, Threads = 4096, pi =   3.1494162400, error = 2.49e-03, time (sec) =   0.1369
Trials = 100000000, Threads = 8192, pi =   3.1377031200, error = 1.24e-03, time (sec) =   0.2737
```

1.1. (10 points) Plot execution time versus p to demonstrate how time varies with the number of threads.

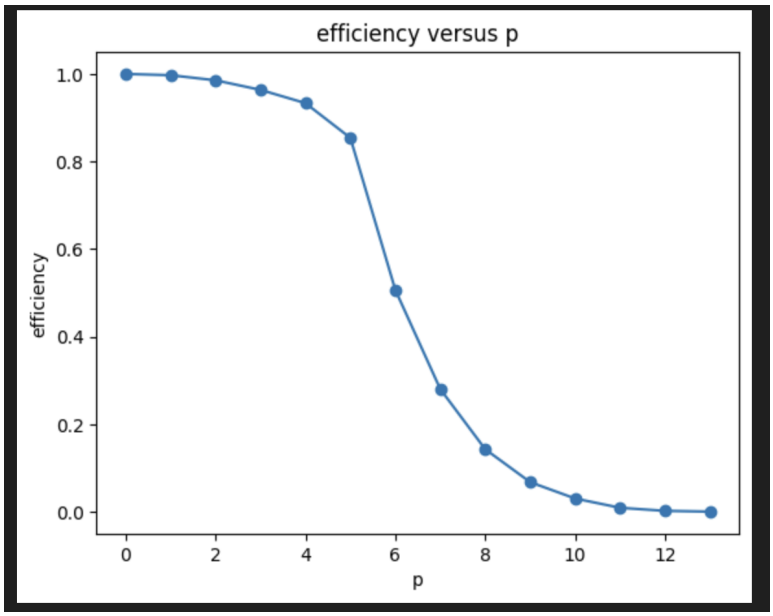


1.2. (10 points) Plot speedup versus p to demonstrate the change in speedup with p.
Speedup = [1.0, 1.994254658385093, 3.9419889502762433, 7.708883553421368, 14.933720930232559, 27.325531914893617, 32.43181818181818, 35.874301675977655, 36.58974358974359, 34.71081081081081, 31.17233009708738, 18.998520710059175, 9.381300219138057, 4.6923639020825725]



1.3. (5 points) Using the definition: efficiency = speedup/p, plot efficiency versus p to demonstrate how efficiency changes as the number of threads are increased.

Efficiency = [1.0, 0.9971273291925465, 0.9854972375690608, 0.963610444177671, 0.9333575581395349, 0.8539228723404255, 0.5067471590909091, 0.28026798184357543, 0.1429286858974359, 0.06779455236486487, 0.030441728610436893, 0.009276621440458582, 0.002290356498813002, 0.000572798327890939]



1.4. (5 points) In your experiments, what value of p minimizes the parallel runtime?
ANS: in my experiments, when p is 256 and k = 8 minimizes the parallel runtime.

2. Repeat the experiments with n=10^10 to obtain the execution time for p=2k, for k = 0, 1, …, 13.

```
Trials = 10000000000, Threads =    1, pi =   1.4236202260, error = 5.47e-01, time (sec) = 127.5424
Trials = 10000000000, Threads =    2, pi =   3.1416070092, error = 4.57e-06, time (sec) =  63.9059
Trials = 10000000000, Threads =    4, pi =   3.1416060276, error = 4.26e-06, time (sec) =  31.9454
Trials = 10000000000, Threads =    8, pi =   3.1416118228, error = 6.10e-06, time (sec) =  15.9876
Trials = 10000000000, Threads =   16, pi =   3.1416066896, error = 4.47e-06, time (sec) =   8.0048
Trials = 10000000000, Threads =   32, pi =   3.1416332068, error = 1.29e-05, time (sec) =   4.0086
Trials = 10000000000, Threads =   64, pi =   3.1416139092, error = 6.77e-06, time (sec) =   3.2660
Trials = 10000000000, Threads =  128, pi =   3.1415943424, error = 5.38e-07, time (sec) =   2.8004
Trials = 10000000000, Threads =  256, pi =   3.1416473120, error = 1.74e-05, time (sec) =   2.7011
Trials = 10000000000, Threads =  512, pi =   3.1415762012, error = 5.24e-06, time (sec) =   2.6941
Trials = 10000000000, Threads = 1024, pi =   3.1414319268, error = 5.12e-05, time (sec) =   2.6863
Trials = 10000000000, Threads = 2048, pi =   3.1412588724, error = 1.06e-04, time (sec) =   2.7007
Trials = 10000000000, Threads = 4096, pi =   3.1407169720, error = 2.79e-04, time (sec) =   2.7164
Trials = 10000000000, Threads = 8192, pi =   3.1412633928, error = 1.05e-04, time (sec) =   2.7735
```

2.1. (5 points) In this case, what value of p minimizes the parallel runtime?
ANS: in my experiments, when p is 1024 minimizes the parallel runtime.

2.2. (5 points) Do you expect the runtime to increase as p is increased beyond a certain value? If so, why? And is this observed in your experiments.

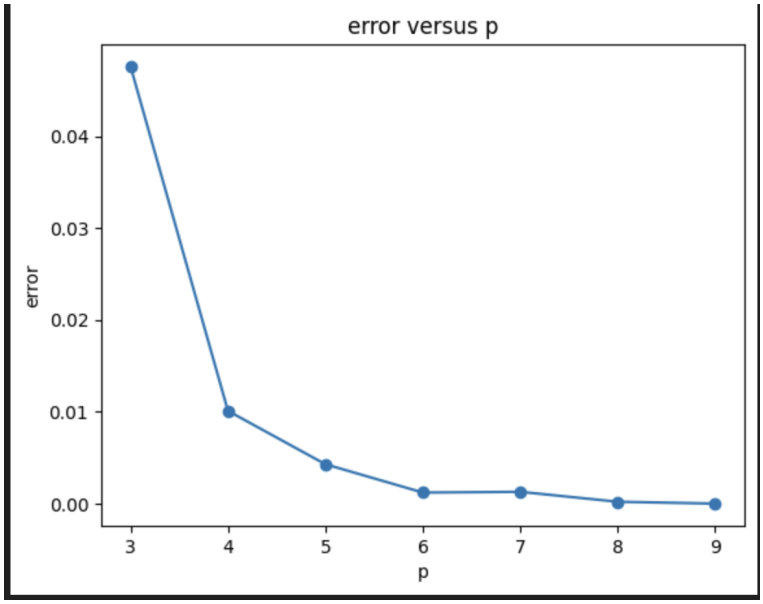
Yes, I expect the runtime to increase as p is increased beyond a certain value. There are two main issues will affect parallelism: partitioning and communication. Above two cases, we can see p = 256 which is best for n = 10^8, and p = 1024 which is best for n = 10^10. I think it will cause the issue because of limit number of tasks. Therefore, it will execute the task sequentially.

3. (5 points) Do you expect that there would be a difference in the number of threads needed to obtain the minimum execution time for two values of n? Is this observed in your experiments.

Yes, I expect that there would be a difference in the number of threads needed to obtain the minimum execution time for two values of n. The different n value and thread p value will get the different execution time base on the problem 1 and problem 2 graph. When we increase threads, more tasks can be parallelized and can obtain the minimum execution time.

4. (5 points) Plot error versus n to illustrate accuracy of the algorithm as a function of n. You may have to run experiments with different values of n; for example n could be chosen to be 10_k, for k = 3, …, 9. Use p = 48.

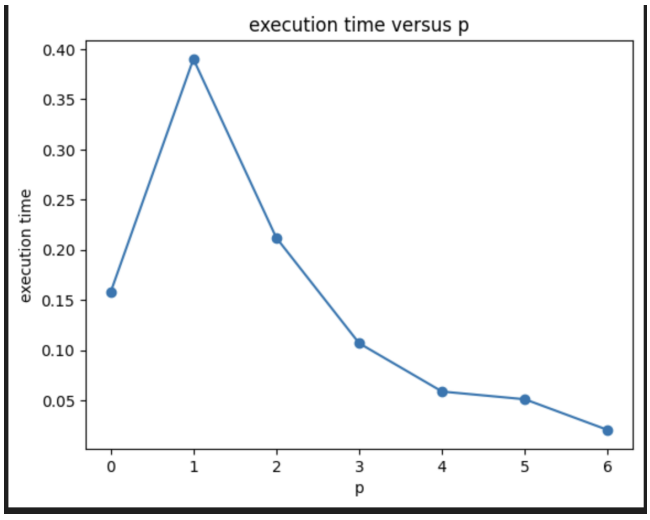
```
Trials = 1000, Threads =   48, pi =   2.9920000000, error = 4.76e-02, time (sec) =   0.0026
Trials = 10000, Threads =   48, pi =   3.1732000000, error = 1.01e-02, time (sec) =   0.0017
Trials = 100000, Threads =   48, pi =   3.1550800000, error = 4.29e-03, time (sec) =   0.0018
Trials = 1000000, Threads =   48, pi =   3.1453960000, error = 1.21e-03, time (sec) =   0.0020
Trials = 10000000, Threads =   48, pi =   3.1456372000, error = 1.29e-03, time (sec) =   0.0068
Trials = 100000000, Threads =   48, pi =   3.1409400400, error = 2.08e-04, time (sec) =   0.0325
Trials = 1000000000, Threads =   48, pi =   3.1416179080, error = 8.04e-06, time (sec) =   0.2802
```



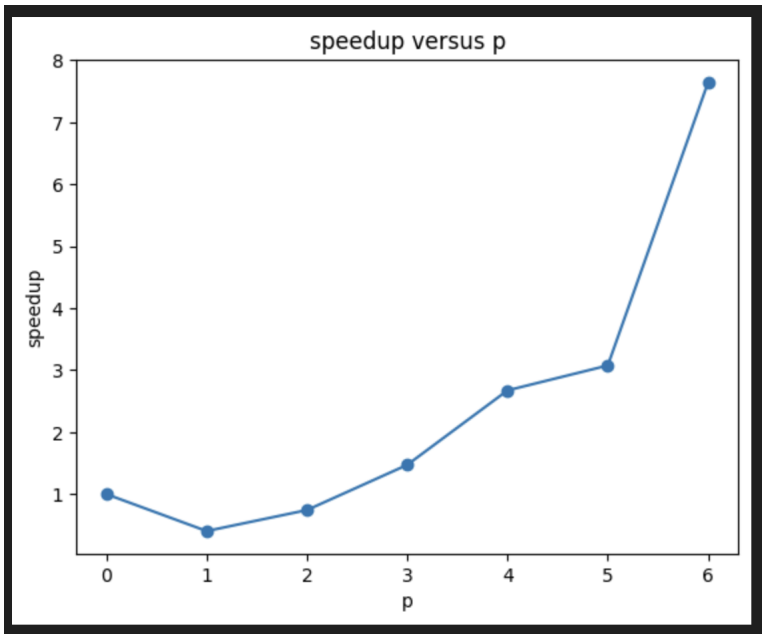
5. Execute the code for n=108 with p chosen to be 2k, for k = 0, 1, …, 6. Specify ntasks-per-node=4 in the job file. Using the experimental data obtained from these experiments, answer the following questions. For plots, use a logarithmic scale for the x-axis.

```
Processes = 1
n = 100000000, p = 1, pi = 3.1415926535904264, relative error = 2.02e-13, time (sec) =   0.1575
Processes = 2
n = 100000000, p = 2, pi = 3.1415926535900223, relative error = 7.29e-14, time (sec) =   0.3901
Processes = 4
n = 100000000, p = 4, pi = 3.1415926535902168, relative error = 1.35e-13, time (sec) =   0.2121
Processes = 8
n = 100000000, p = 8, pi = 3.1415926535896137, relative error = 5.71e-14, time (sec) =   0.1070
Processes = 16
n = 100000000, p = 16, pi = 3.1415926535897754, relative error = 5.65e-15, time (sec) =   0.0589
Processes = 32
n = 100000000, p = 32, pi = 3.1415926535897736, relative error = 6.22e-15, time (sec) =   0.0512
Processes = 64
n = 100000000, p = 64, pi = 3.1415926535897940, relative error = 2.83e-16, time (sec) =   0.0206
```

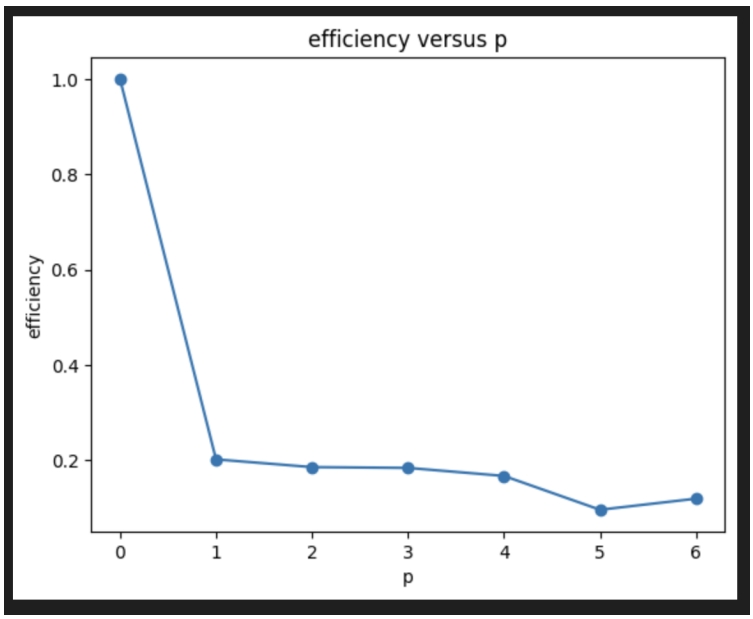
5.1. (10 points) Plot execution time versus p to demonstrate how time varies with the number of processes.



5.2. (10 points) Plot speedup versus p to demonstrate the change in speedup with p.
Speedup = [[1.0, 0.40374263009484745, 0.7425742574257426, 1.47196261682243, 2.67402376910017, 3.076171875, 7.645631067961165]]

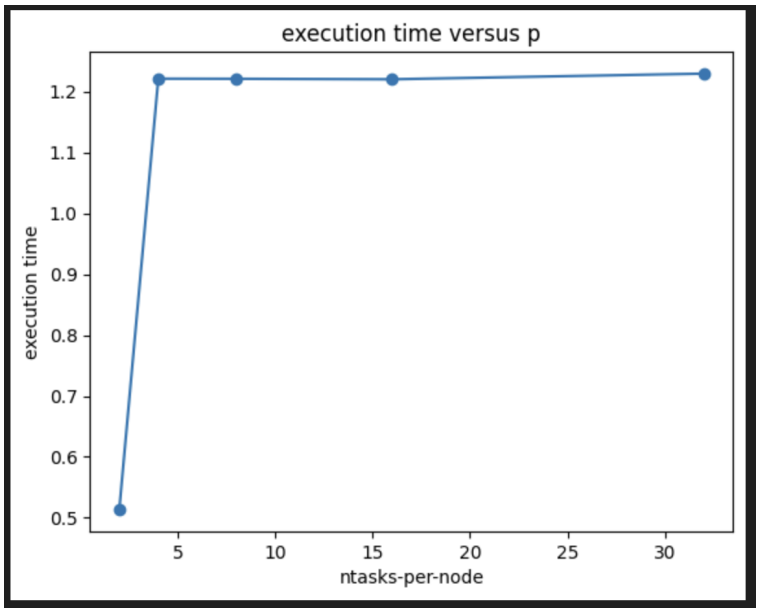


5.3. (5 points) Using the definition: efficiency = speedup/p, plot efficiency versus p to demonstrate how efficiency changes as the number of processes is increased.
Efficiency =[[1.0, 0.20187131504742373, 0.18564356435643564, 0.18399532710280375, 0.1671264855687606, 0.09613037109375, 0.11946298543689321]]



5.4. (5 points) What value of p minimizes the parallel runtime?
When value of p is 64, it minimizes the parallel runtime.

6. (10 points) With n=1010 and p=64, determine the value of ntasks-per-node that minimizes the total_time. Plot time versus ntasks-per-node to illustrate your experimental results for this question.
When ntasks-per-node is 2, it minimizes the total time.
Ntasks-per-node:2,4,8,16,32

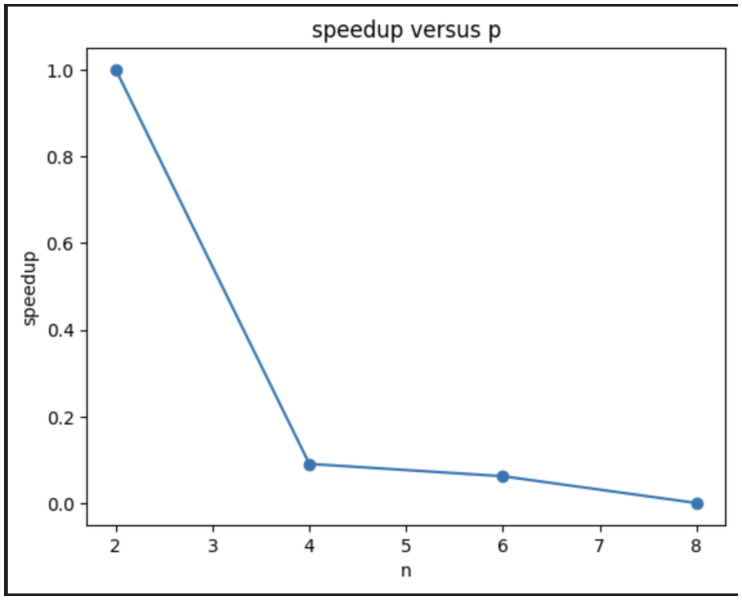


7. Execute the code with p=64 for n=10^2, 10^4, 10^6 and 10^8, with ntasks-per-node=4.

```
10^2
n = 100, p = 64, pi = 3.1416009869231249, relative error = 2.65e-06, time (sec) =   0.0033
10^4
n = 10000, p = 64, pi = 3.1415926544231265, relative error = 2.65e-10, time (sec) =   0.0017
10^6
n = 1000000, p = 64, pi = 3.1415926535898753, relative error = 2.62e-14, time (sec) =   0.0040
10^8
n = 100000000, p = 64, pi = 3.1415926535897940, relative error = 2.83e-16, time (sec) =   0.0151
```

7.1. (5 points) Plot the speedup observed as a function of n on p=64 w.r.t. p=1. You will need to obtain execution time on p=1 for n=10^2, 10^4, 10^6 and 10^8.

```
10^2
n = 100, p = 1, pi = 3.1416009869231254, relative error = 2.65e-06, time (sec) =   0.0001
10^4
n = 10000, p = 1, pi = 3.1415926544231341, relative error = 2.65e-10, time (sec) =   0.0011
10^6
n = 1000000, p = 1, pi = 3.1415926535897643, relative error = 9.19e-15, time (sec) =   0.0016
10^8
n = 100000000, p = 1, pi = 3.1415926535904264, relative error = 2.02e-13, time (sec) =   0.1542
```



7.2. (5 points) Plot the relative error versus n to illustrate the accuracy of the algorithm as a function of n.