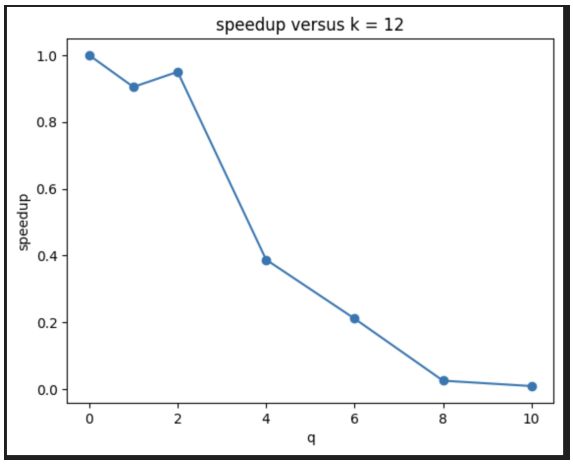
Chonglin Zhang
CSCE-735
DUE: 10-2-2023
833003072

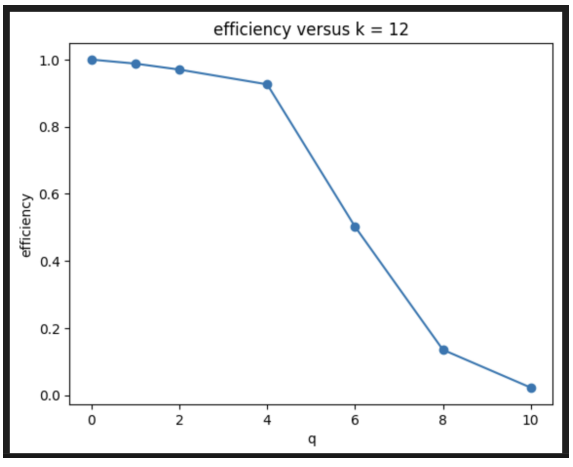
1.

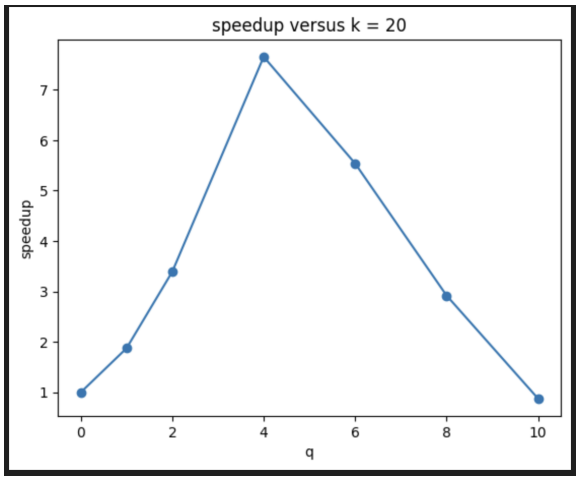
```
List Size = 16, Threads = 2, error = 0, time (sec) =   0.0004, qsort_time =   0.0000
List Size = 16, Threads = 4, error = 0, time (sec) =   0.0006, qsort_time =   0.0000
List Size = 16, Threads = 8, error = 0, time (sec) =   0.0010, qsort_time =   0.0000
List Size = 1048576, Threads = 16, error = 0, time (sec) =   0.0237, qsort_time =   0.1771
List Size = 16777216, Threads = 256, error = 0, time (sec) =   0.2896, qsort_time =   3.3115
```

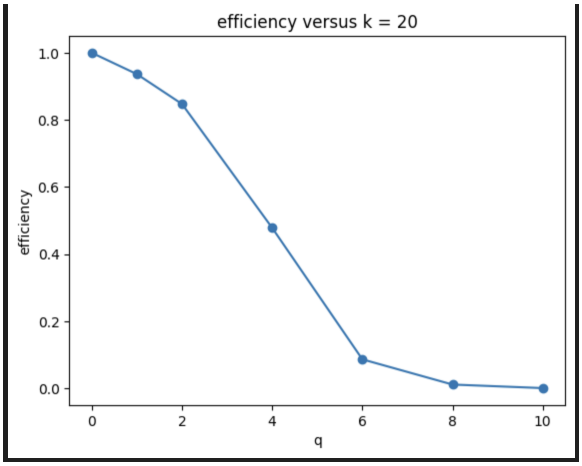
2.

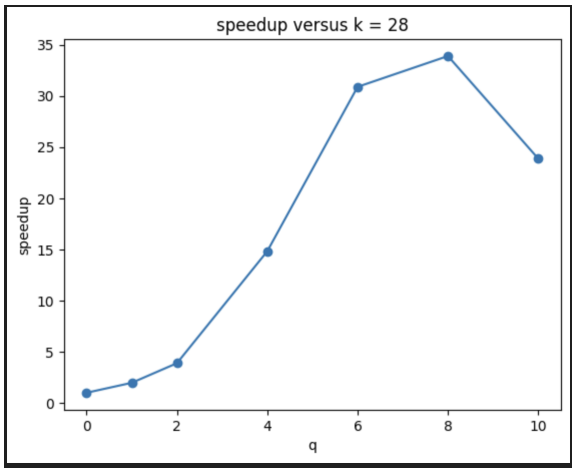
```
k = 12
List Size = 4096, Threads = 1, error = 0, time (sec) =   0.0019, qsort_time =   0.0010
List Size = 4096, Threads = 2, error = 0, time (sec) =   0.0021, qsort_time =   0.0005
List Size = 4096, Threads = 4, error = 0, time (sec) =   0.0020, qsort_time =   0.0006
List Size = 4096, Threads = 16, error = 0, time (sec) =   0.0049, qsort_time =   0.0006
List Size = 4096, Threads = 64, error = 0, time (sec) =   0.0090, qsort_time =   0.0010
List Size = 4096, Threads = 256, error = 0, time (sec) =   0.0750, qsort_time =   0.0006
List Size = 4096, Threads = 1024, error = 0, time (sec) =   0.2069, qsort_time =   0.0007
k = 20
List Size = 1048576, Threads = 1, error = 0, time (sec) =   0.1783, qsort_time =   0.1762
List Size = 1048576, Threads = 2, error = 0, time (sec) =   0.0952, qsort_time =   0.1736
List Size = 1048576, Threads = 4, error = 0, time (sec) =   0.0526, qsort_time =   0.1734
List Size = 1048576, Threads = 16, error = 0, time (sec) =   0.0233, qsort_time =   0.1772
List Size = 1048576, Threads = 64, error = 0, time (sec) =   0.0322, qsort_time =   0.1721
List Size = 1048576, Threads = 256, error = 0, time (sec) =   0.0611, qsort_time =   0.1740
List Size = 1048576, Threads = 1024, error = 0, time (sec) =   0.2036, qsort_time =   0.1720
k = 28
List Size = 268435456, Threads = 1, error = 0, time (sec) =  63.0775, qsort_time =  62.6952
List Size = 268435456, Threads = 2, error = 0, time (sec) =  31.9805, qsort_time =  62.5950
List Size = 268435456, Threads = 4, error = 0, time (sec) =  16.2315, qsort_time =  62.6684
List Size = 268435456, Threads = 16, error = 0, time (sec) =   4.2559, qsort_time =  62.6667
List Size = 268435456, Threads = 64, error = 0, time (sec) =   2.0429, qsort_time =  62.5664
List Size = 268435456, Threads = 256, error = 0, time (sec) =   1.8615, qsort_time =  62.9973
List Size = 268435456, Threads = 1024, error = 0, time (sec) =   2.6437, qsort_time =  63.1377
```

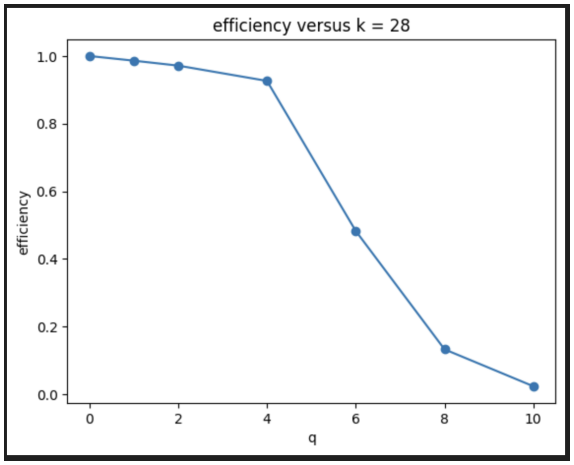








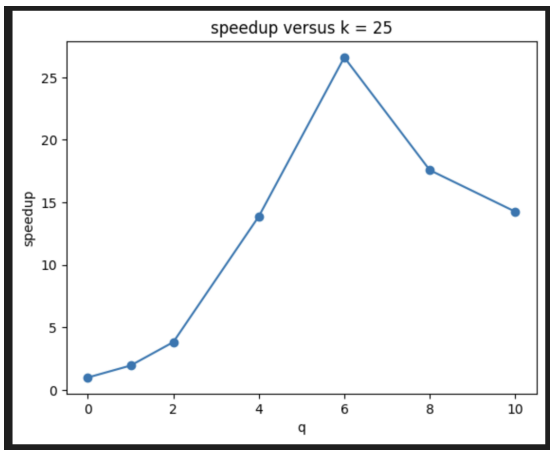


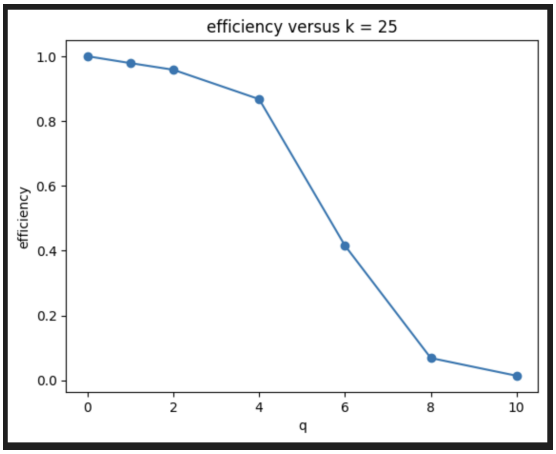


Using these diagram to compare speed up and efficiency. For k = 12, we can see that there are no much improve for the speed up and efficiency. The reason causes that the load may be light which the system can handle it. For k = 20, when q is 4, it is the highest point in the domain in the speed up graph. I think the most q are process, and it may not have idle q for other jobs. For efficiency part, when there are more q, the efficiency is linear decreasing which it may not be best q for k = 20. It might need to wait other to finish job before execute. For k = 28, the speed up graph increase speed rapidly. It means more q can handle and deal with jobs faster. However, it has the linear decreasing graph because q still can handle more jobs.

3

I used the value k when k = 25 and k = 28, and I used the diagram to compare show their speed up. When the k value become bigger than before, the speed up will be faster than before. Therefore, I used k = 25 and k = 28 to compare with each other. We can see speedup and efficiency. Therefore, my code is well design and it has been parallelized.


speedup versus k = 25
speedup
q
0
5
10
15
20
25
0
2
4
6
8
10


efficiency versus k = 25
efficiency
q
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10