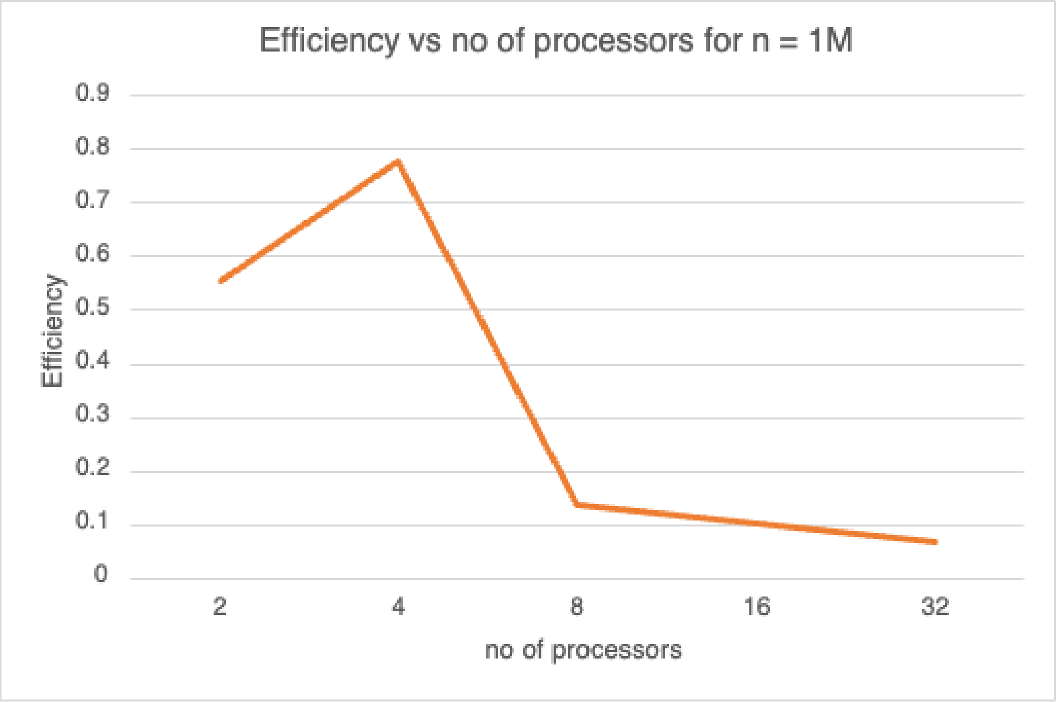
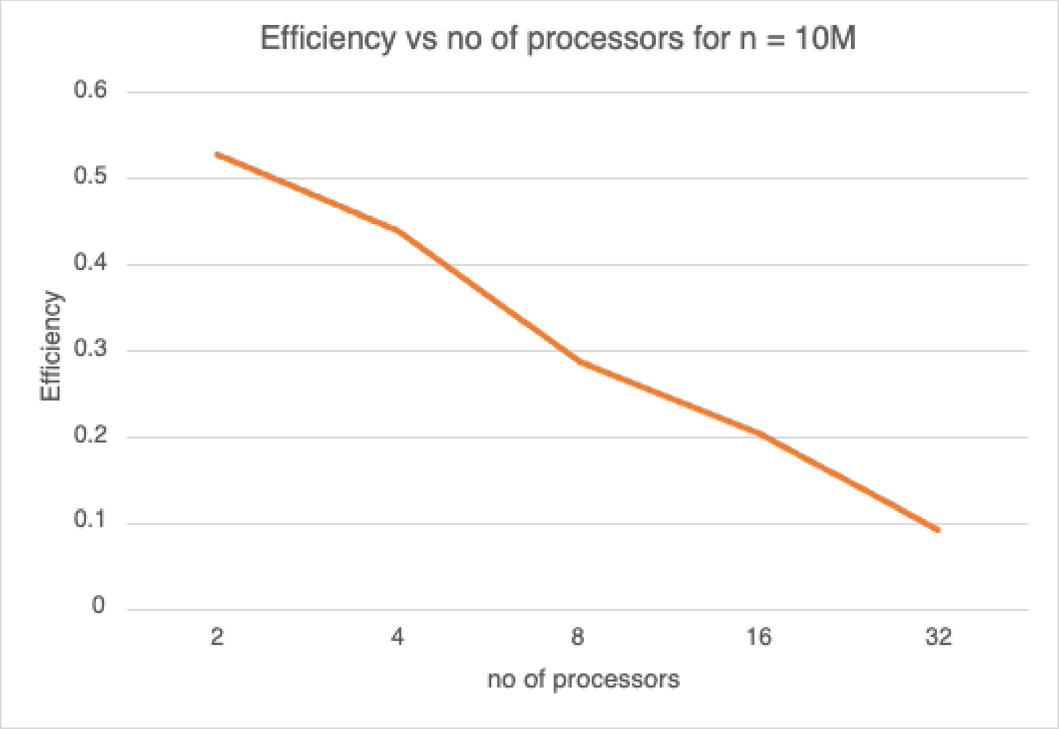
CS525 Parallel Computing HW6

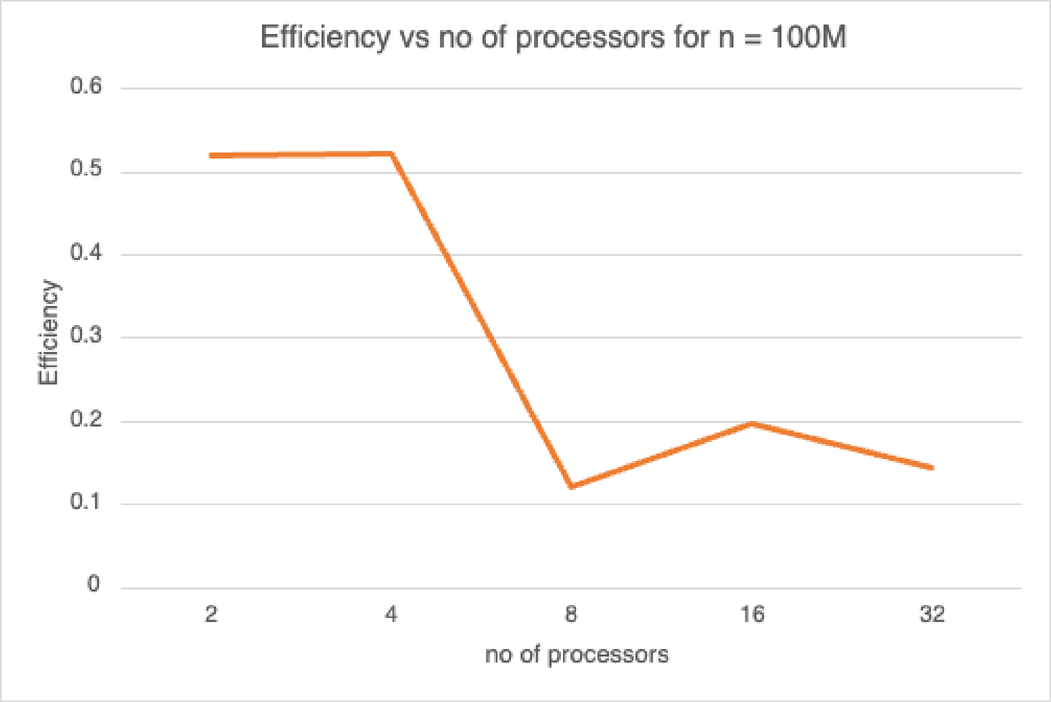
Q1.

1. Constant problem size (W)

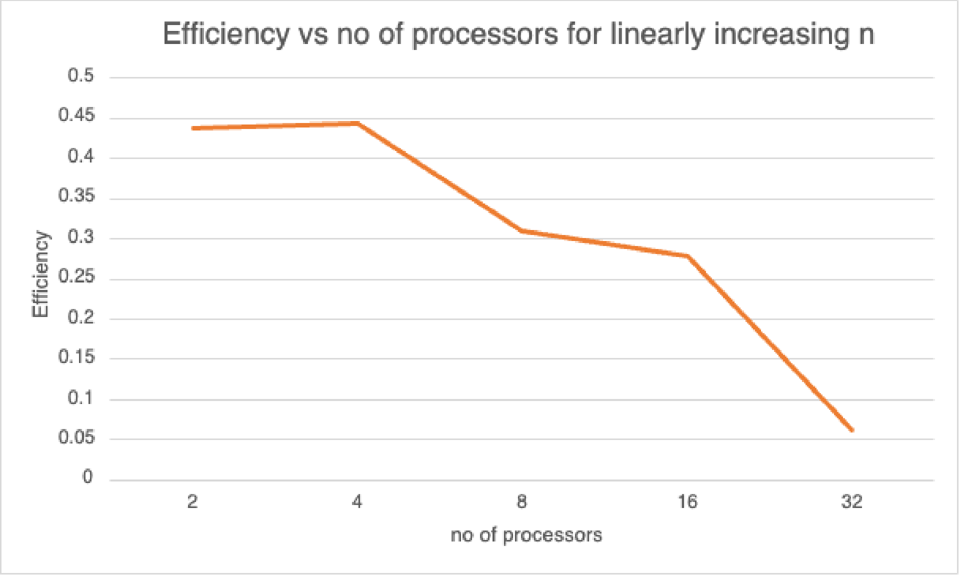
Efficiency calculated as = Serial time / (p * Parallel time)




Efficiency vs no of processors for n = 10M
0.6
0.5
0.4
0.3
0.2
0.1
0
Efficiency
2
4
8
16
32
no of processors


Efficiency vs no of processors for n = 100M
0.6
0.5
0.4
0.3
0.2
0.1
0
Efficiency
2
4
8
16
32
no of processors

2. Linearly increasing number of elements (n)



3. Problem size increasing at the rate of isoefficiency

Problem size W increasing at a rate of isoefficiency

For QuickSort, W, that is, asymptotic number of serial operations are nlogn.

For isoefficiency, W should increase at a rate of **$p\log^2 p$**

Therefore, W ~ $p\log^2 p$.

nlogn ~ $p\log^2 p$

logn ~ $\log^2 p$

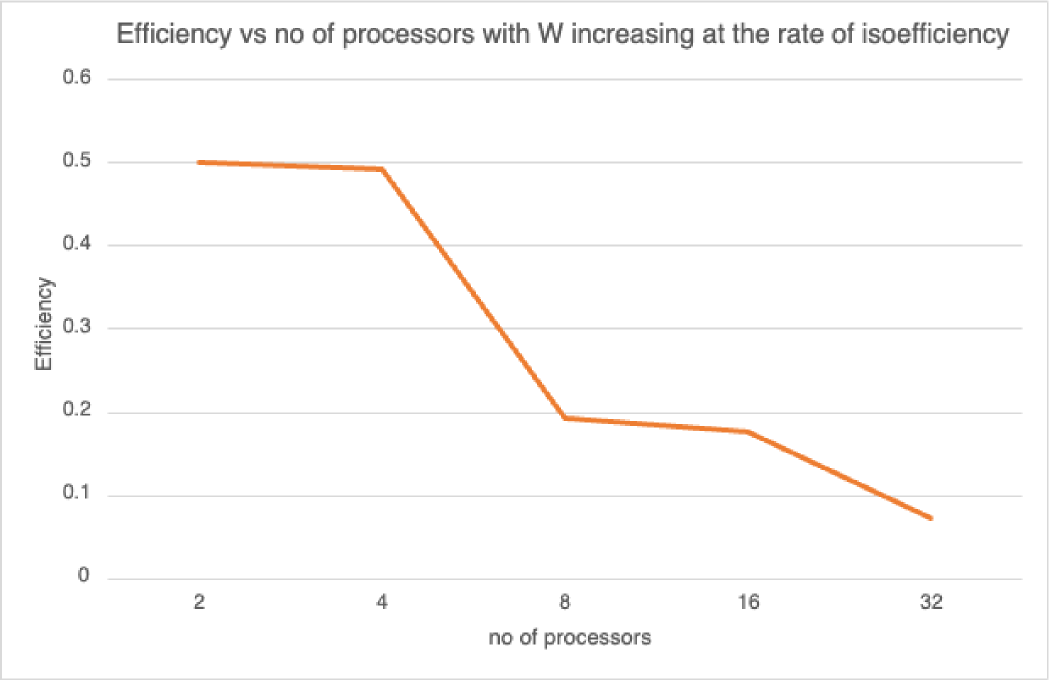
n ~ plogp

n = cplogp , c = n/plogp

Calcuated constant term c by taking n = 1M and p=2 as base, we get c = 500000

Therefore, **n= 500000 plogp**

Calculated values of n with p = {2,4 8,16,32} and plotted below graph



Output of the code: (Disclaimer: The mc machines were overloaded during testing the code, hence might see higher values for time in some cases)

Case 1:

```
Length: 1024000 no of processors:2 parallel time: 0.906581 serial time: 1.00942 Efficiency: 0.55672

Length: 1024000 no of processors:4 parallel time: 0.239332 serial time: 0.745449 Efficiency: 0.778678

Length: 1024000 no of processors:8 parallel time: 0.399754 serial time: 0.445258 Efficiency: 0.139229

Length: 1024000 no of processors:16 parallel time: 0.281038 serial time: 0.455861 Efficiency: 0.101379

Length: 1024000 no of processors:32 parallel time: 0.216882 serial time: 0.463959 Efficiency: 0.0668507

Length: 10240000 no of processors:2 parallel time: 9.82099 serial time: 10.3544 Efficiency: 0.527155

Length: 10240000 no of processors:4 parallel time: 3.24982 serial time: 5.72316 Efficiency: 0.440268

Length: 10240000 no of processors:8 parallel time: 3.09307 serial time: 7.13792 Efficiency: 0.288464

Length: 10240000 no of processors:16 parallel time: 1.88847 serial time: 6.16761 Efficiency: 0.204121

Length: 10240000 no of processors:32 parallel time: 2.26686 serial time: 6.66141 Efficiency: 0.0918314

Length: 102400000 no of processors:2 parallel time: 130.7 serial time: 136.044 Efficiency: 0.520443

Length: 102400000 no of processors:4 parallel time: 41.3972 serial time: 86.3916 Efficiency: 0.521724

Length: 102400000 no of processors:8 parallel time: 70.6472 serial time: 68.4937 Efficiency: 0.12119
```

```
Length: 102400000 no of processors:16 parallel time: 25.726 serial time: 80.9599 Efficiency: 0.196688
Length: 102400000 no of processors:32 parallel time: 13.082 serial time: 60.2371 Efficiency: 0.143893
```

Case 2:

```
Length: 1000000 no of processors:2 parallel time: 0.518631 serial time: 0.453843 Efficiency: 0.43754
Length: 2000000 no of processors:4 parallel time: 0.564898 serial time: 1.0023 Efficiency: 0.443576
Length: 3000000 no of processors:8 parallel time: 0.703205 serial time: 1.74583 Efficiency: 0.310335
Length: 4000000 no of processors:16 parallel time: 0.454677 serial time: 2.02642 Efficiency: 0.278551
Length: 5000000 no of processors:32 parallel time: 1.30626 serial time: 2.6132 Efficiency: 0.0625164
```

Case 3:

```
Length: 997920 no of processors:2 parallel time: 0.535968 serial time: 0.536341 Efficiency: 0.500348
Length: 4001760 no of processors:4 parallel time: 1.4652 serial time: 2.8873 Efficiency: 0.492647
Length: 12000240 no of processors:8 parallel time: 4.71856 serial time: 7.25958 Efficiency: 0.192315
Length: 31998960 no of processors:16 parallel time: 6.72807 serial time: 19.0227 Efficiency: 0.17671
Length: 79999920 no of processors:32 parallel time: 21.2135 serial time: 49.1305 Efficiency: 0.0723749
```

Execute the code:

You need to run the script **run.sh**, which will be found in hw6/question1/ folder

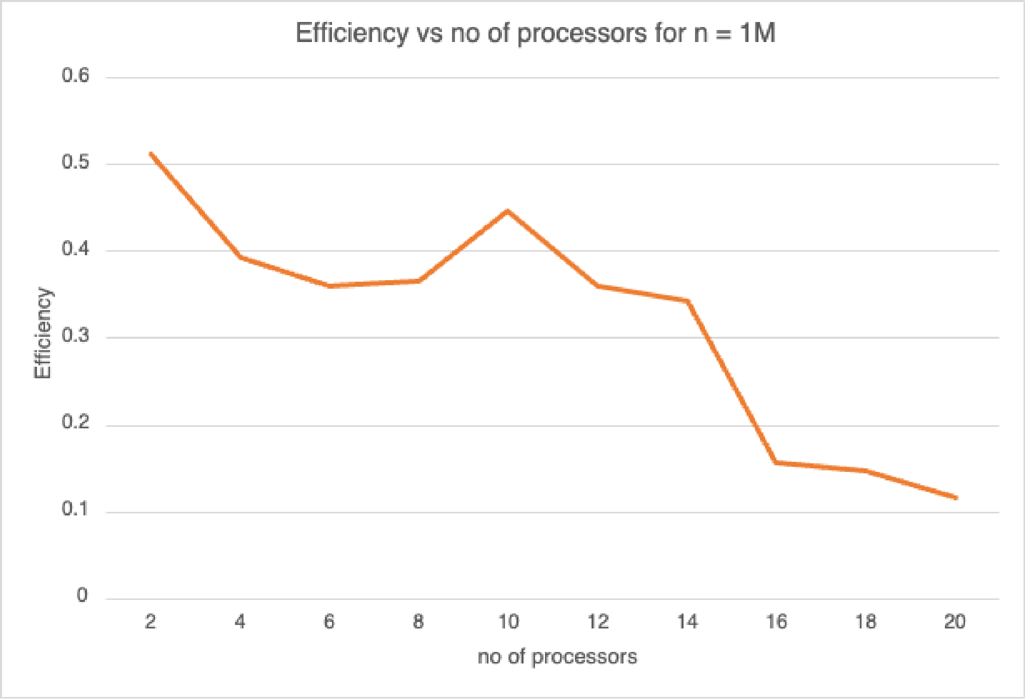
The **script itself has compile and execute step :**

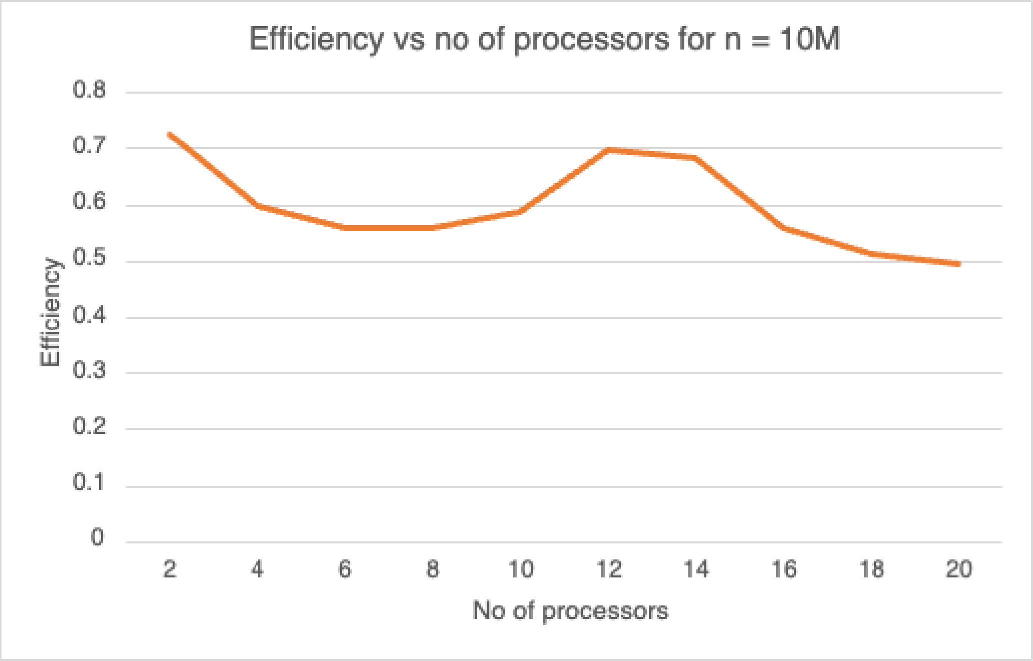
```
mpic++ -o quicksort_p1 quicksort_p1.cpp
```

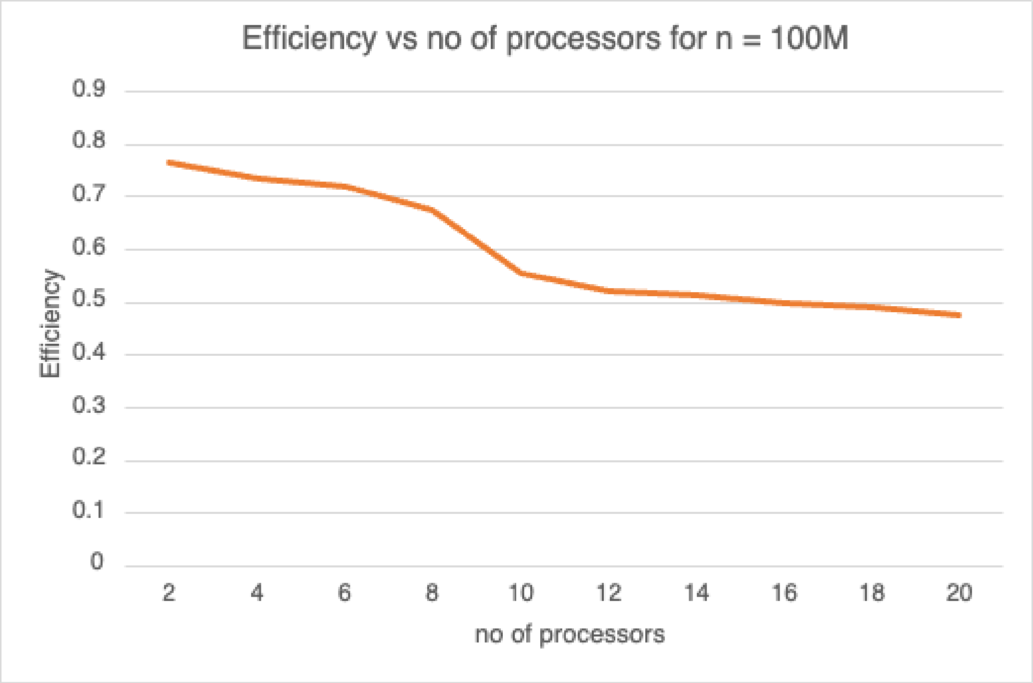
Execute the script: **./run.sh**

Q2.

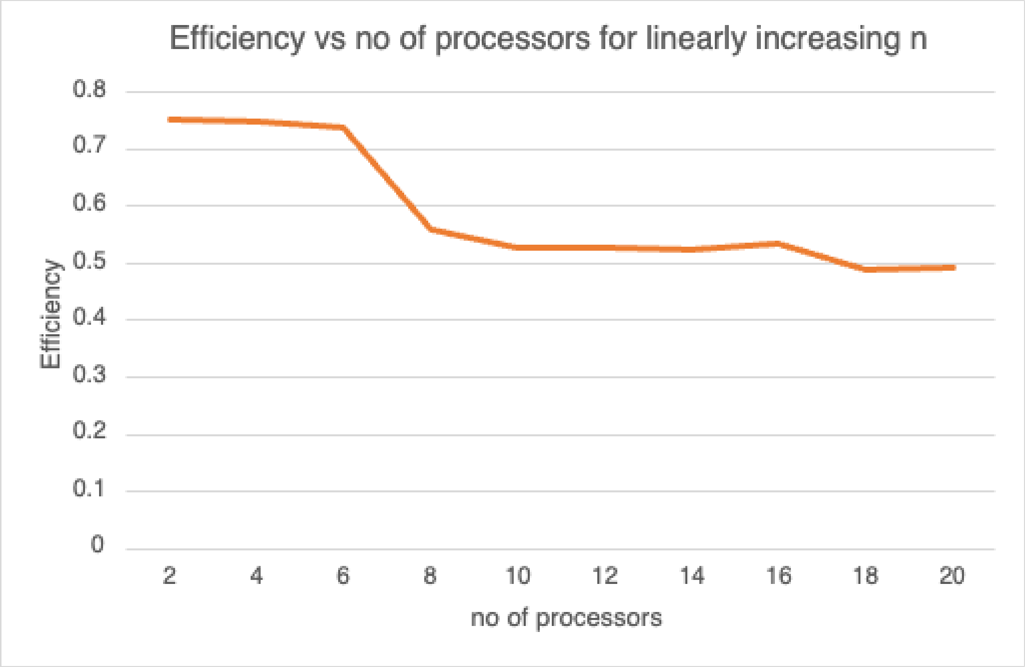
1. Constant problem size (W)







2. Linearly increasing number of elements (n)



3. Problem size increasing at the rate of isoefficiency

Problem size W increasing at a rate of isoefficiency

For Sample sort, W, that is, asymptotic number of serial operations are nlogn (have used quick sort, hence nlogn)

For isoefficiency, W should increase at a rate of p^3 logp

Therefore, W ~ p^3 logp.

nlogn ~ p^3 logp

Applying log on both sides we get,

logn ~ log p^3

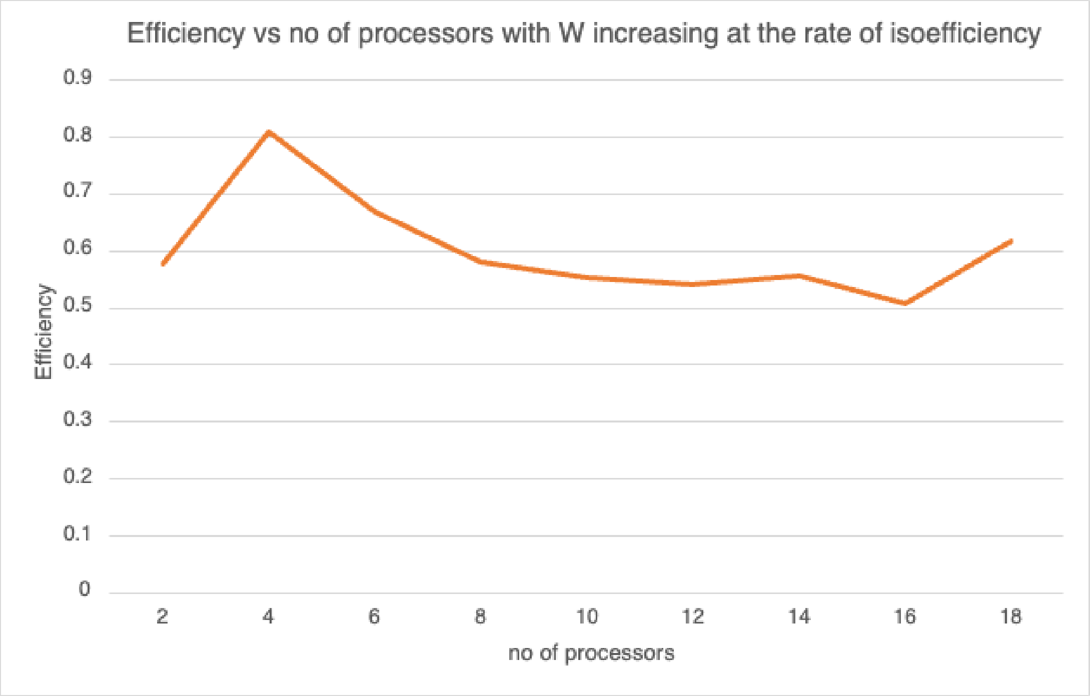
n ~ p^3

n = c p^3 , c = n / p^3

Calcuated constant term c by taking n = 1M and p=2 as base, we get c =125000

Therefore, **n= 125000 p^3**

Calculated values of n with p = {2,4 , 6,8, 10,12,14,16,18,20} and plotted below graph



Output of the Code: (Disclaimer: The mc machines were overloaded during testing the code, hence might see higher values for time in some cases)

Case 1:

```
Length: 1024000 no of processors 2 parallel time: 0.806957 serial time: 0.826904 Efficiency: 0.51236
Length: 1024000 no of processors 4 parallel time: 0.222977 serial time: 0.350096 Efficiency: 0.392525
Length: 1024000 no of processors 6 parallel time: 0.161702 serial time: 0.349689 Efficiency: 0.360426
Length: 1024000 no of processors 8 parallel time: 0.118299 serial time: 0.346495 Efficiency: 0.366123
Length: 1024000 no of processors 10 parallel time: 0.0944985 serial time: 0.422443 Efficiency: 0.447037
Length: 1024000 no of processors 12 parallel time: 0.0821794 serial time: 0.356039 Efficiency: 0.361038
Length: 1024000 no of processors 14 parallel time: 0.0724315 serial time: 0.347767 Efficiency: 0.342952
Length: 1024000 no of processors 16 parallel time: 0.148349 serial time: 0.374513 Efficiency: 0.157784
Length: 1024000 no of processors 18 parallel time: 0.178137 serial time: 0.473195 Efficiency: 0.147575
Length: 1024000 no of processors 20 parallel time: 0.192137 serial time: 0.446763 Efficiency: 0.116262
Length: 10240000 no of processors 2 parallel time: 1.91698 serial time: 2.78408 Efficiency: 0.726161
Length: 10240000 no of processors 4 parallel time: 1.13941 serial time: 2.7205 Efficiency: 0.596912
Length: 10240000 no of processors 6 parallel time: 0.837179 serial time: 2.81434 Efficiency: 0.560283
Length: 10240000 no of processors 8 parallel time: 0.639395 serial time: 2.86092 Efficiency: 0.559302
Length: 10240000 no of processors 10 parallel time: 0.490596 serial time: 2.89128 Efficiency: 0.58934
Length: 10240000 no of processors 12 parallel time: 0.343309 serial time: 2.87989 Efficiency: 0.699051
Length: 10240000 no of processors 14 parallel time: 0.301483 serial time: 2.88238 Efficiency: 0.682905
Length: 10240000 no of processors 16 parallel time: 0.321562 serial time: 2.87509 Efficiency: 0.558814
Length: 10240000 no of processors 18 parallel time: 0.311884 serial time: 2.88444 Efficiency: 0.513802
Length: 10240000 no of processors 20 parallel time: 0.294343 serial time: 2.92434 Efficiency: 0.496758
Length: 102400000 no of processors 2 parallel time: 21.6757 serial time: 33.2251 Efficiency: 0.766414
Length: 102400000 no of processors 4 parallel time: 11.3463 serial time: 33.4592 Efficiency: 0.737226
Length: 102400000 no of processors 6 parallel time: 7.77786 serial time: 33.5608 Efficiency: 0.719152
Length: 102400000 no of processors 8 parallel time: 6.21975 serial time: 33.5566 Efficiency: 0.674397
Length: 102400000 no of processors 10 parallel time: 5.99863 serial time: 33.3255 Efficiency: 0.555552
Length: 102400000 no of processors 12 parallel time: 5.36318 serial time: 33.6003 Efficiency: 0.522083
```

```
Length: 102400000 no of processors 14 parallel time: 4.68751 serial time: 33.8304 Efficiency: 0.51551
Length: 102400000 no of processors 16 parallel time: 4.22986 serial time: 33.8704 Efficiency: 0.500466
Length: 102400000 no of processors 18 parallel time: 3.84374 serial time: 33.9942 Efficiency: 0.491336
Length: 102400000 no of processors 20 parallel time: 3.56559 serial time: 34.0211 Efficiency: 0.477075
```

Case 2:

```
Length: 1000000 no  of  processors  2 parallel  time: 0.156187  serial  time: 0.234797  Efficiency: 0.751654
Length: 2000000 no  of  processors  4 parallel  time: 0.166858  serial  time: 0.499332  Efficiency: 0.748136
Length: 3000000 no  of  processors  6 parallel  time: 0.17401 serial  time: 0.771628  Efficiency: 0.739064
Length: 4000000 no  of  processors  8 parallel  time: 0.235531  serial  time: 1.05455 Efficiency: 0.559668
Length: 5000000 no  of  processors  10  parallel  time: 0.25619 serial  time: 1.35334 Efficiency: 0.528257
Length: 6000000 no  of  processors  12  parallel  time: 0.257341  serial  time: 1.62678 Efficiency: 0.526793
Length: 7000000 no  of  processors  14  parallel  time: 0.263468  serial  time: 1.9388  Efficiency: 0.525624
Length: 8000000 no  of  processors  16  parallel  time: 0.27187 serial  time: 2.33036 Efficiency: 0.535726
Length: 9000000 no  of  processors  18  parallel  time: 0.292793  serial  time: 2.57003 Efficiency: 0.487647
Length: 10000000  no  of  processors  20  parallel  time: 0.287692  serial  time: 2.84303 Efficiency: 0.49411
```

Case 3:

```
Length: 997920 no of processors 2 parallel time: 0.20249 serial time: 0.234378 Efficiency: 0.57874
Length: 7998480 no of processors 4 parallel time: 0.724929 serial time: 2.34775 Efficiency: 0.809649
Length: 26999280 no of processors 6 parallel time: 2.03376 serial time: 8.16666 Efficiency: 0.669258
Length: 63997920 no of processors 8 parallel time: 4.42321 serial time: 20.5921 Efficiency: 0.581934
Length: 125002080 no of processors 10 parallel time: 7.5701 serial time: 41.8411 Efficiency: 0.552716
```

```
Length: 215999280 no of processors 12 parallel time: 11.4466 serial time: 74.4612 Efficiency: 0.542092
Length: 343002240 no of processors 14 parallel time: 15.5185 serial time: 120.792 Efficiency: 0.555981
Length: 511998480 no of processors 16 parallel time: 22.5946 serial time: 184.048 Efficiency: 0.509103
Length: 729000720 no of processors 18 parallel time: 24.0928 serial time: 267.327 Efficiency: 0.616429
```

Execute the code:

You need to run the script **run.sh**, which will be found in hw6/question2/ folder

The **script itself has compile and execute step**:

```
mpic++ -o bucketsort_p1 bucketsort_p1.cpp
```

Execute the script: **./run.sh**

If you get permission denied running the script, please set **chmod +x run.sh** and then execute ./run.sh