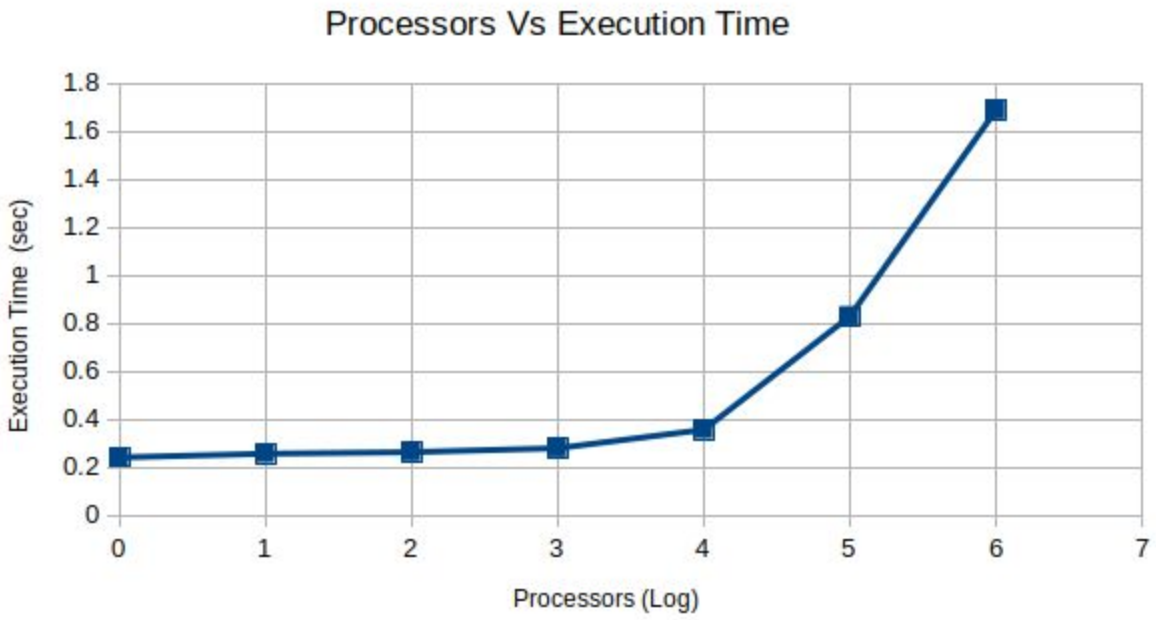
GAUTHAM SRINIVASAN
927008557
CSCE 689 HW4

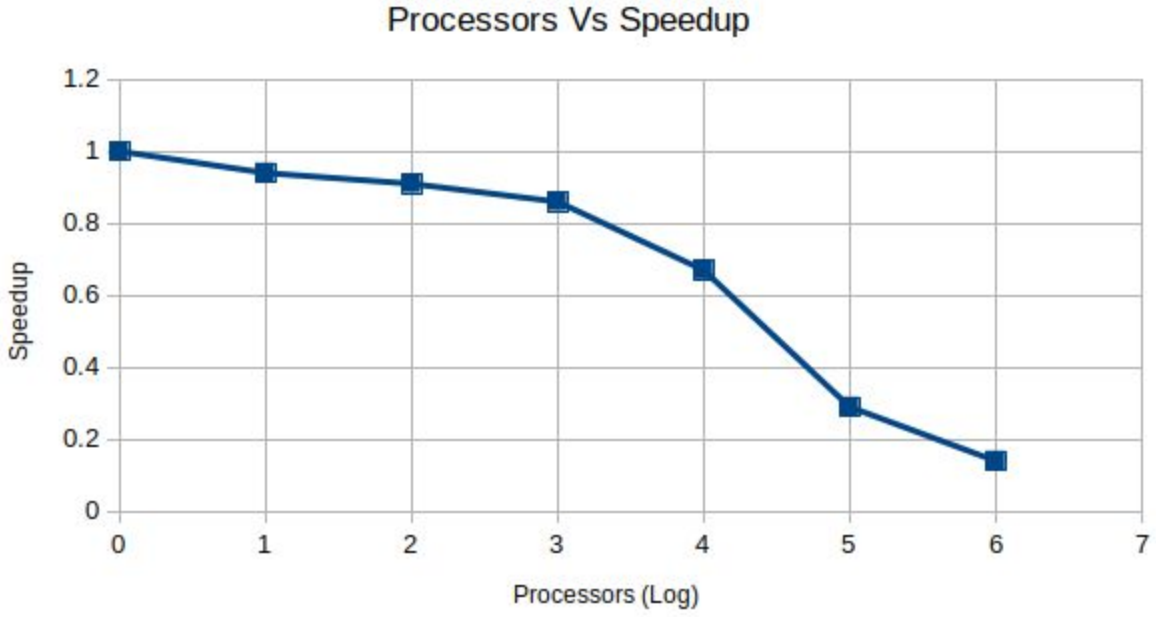
1. **Complete the MPI-based code provided in qsort_hypercube.c to implement the parallel quicksort algorithm for a d-dimensional hypercube with p=2 d processors. 60 points will be awarded if the code compiles and executes the following command successfully.**

   Attached code snippet *qsort_hypercube.c and qsort_hypercube.h*

2. **Weak Scalability Study: Run your code to sort a distributed list of size np where n is the size of the local list on each process and p is the number of processes. For your experiments, use n=20,480,000 and p = 1, 2, 4, 8, 16, 32, and 64. Set type=0. Plot the execution time, speedup, and efficiency of your code as a function of p. Use a logarithmic scale for the x-axis.**

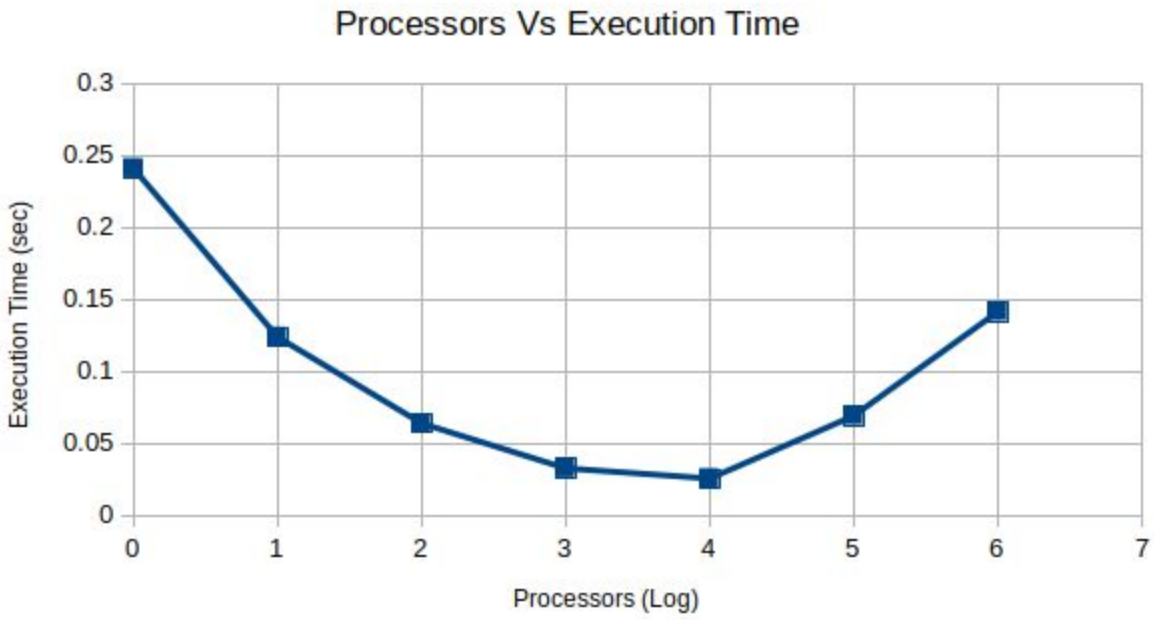
Processors	Processors (Log)	Time (sec)	Speedup	Efficiency
1	0	0.241711	1	1
2	1	0.256959	0.94	0.47
4	2	0.263782	0.91	0.23
8	3	0.280026	0.86	0.11
16	4	0.356910	0.67	0.04
32	5	0.829137	0.29	0.01
64	6	1.687360	0.14	0.002

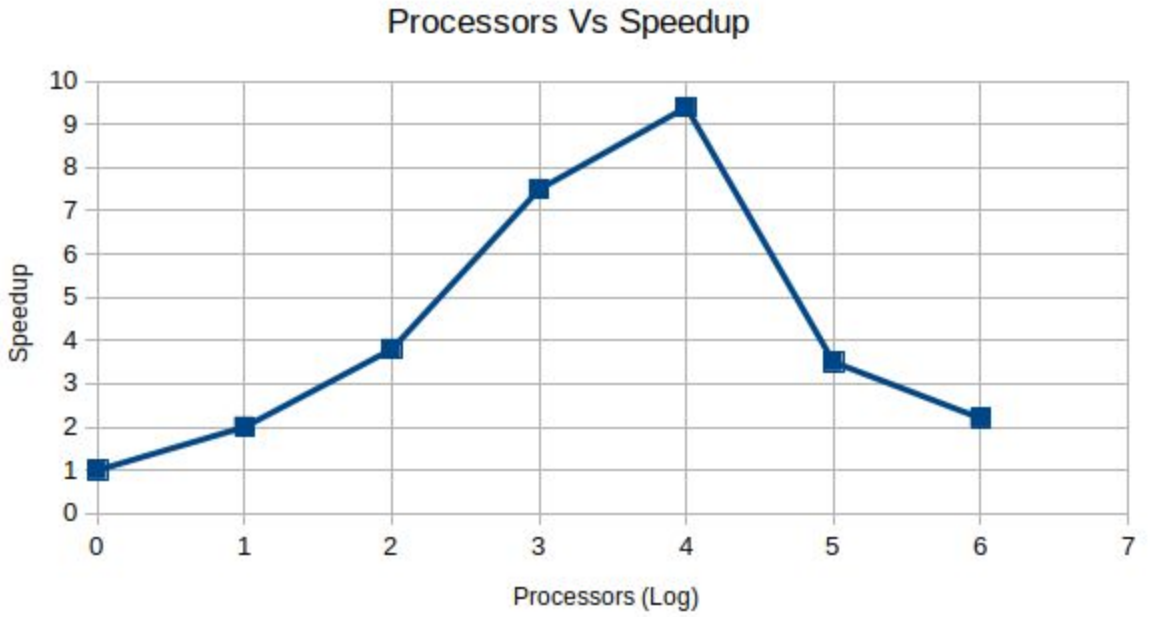

Processors Vs Execution Time
Execution Time (sec)
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
Processors (Log)

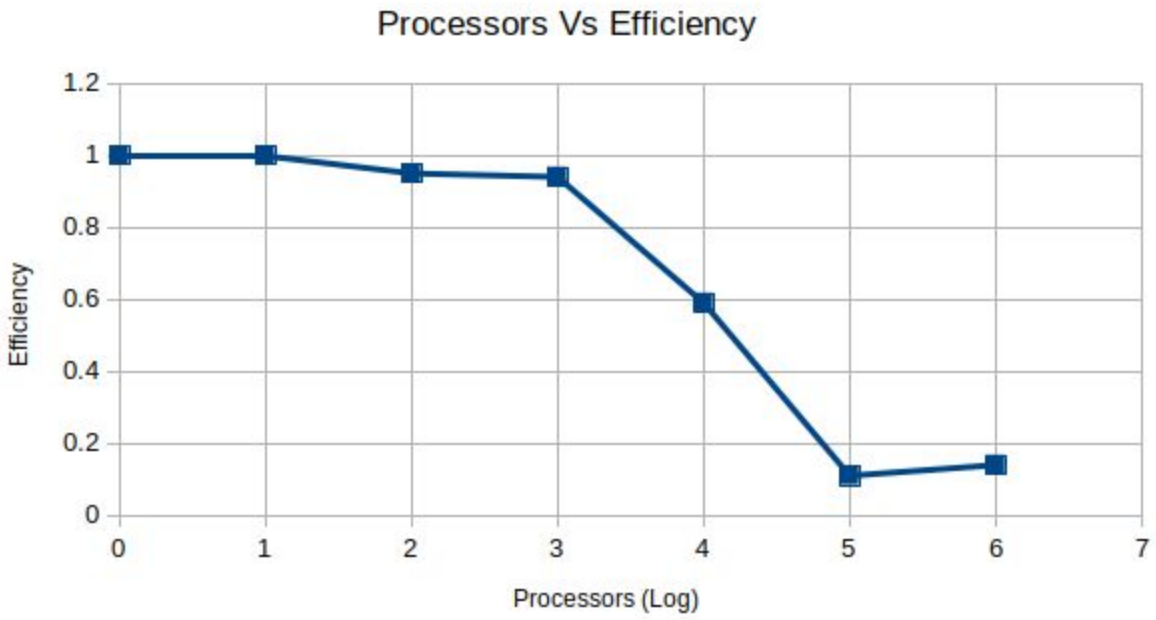

Processors Vs Speedup
Speedup
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
Processors (Log)

3. **Strong Scalability Study: Now run your code with n=20,480,000/p where p = 1, 2, 4, 8, 16, 32, and 64. Set type=0. Plot the execution time, speedup, and efficiency of your code as a function of p. Use a logarithmic scale for the x-axis.**

Processors	Processors (Log)	Time (sec)	Speedup	Efficiency
1	0	0.240959	1	1
2	1	0.123826	2	1
4	2	0.063871	3.8	0.95
8	3	0.032730	7.5	0.94
16	4	0.025654	9.4	0.59
32	5	0.069190	3.5	0.11
64	6	0.141512	2.2	0.14


Processors Vs Execution Time
Execution Time (sec)
0.3
0.25
0.2
0.15
0.1
0.05
0
0
1
2
3
4
5
6
7
Processors (Log)


Processors Vs Speedup
Speedup
10
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
Processors (Log)



4. **Modify the code to sort the list in descending order. Submit the modified code as qsort_hypercube_descending.c. 2 points will be awarded for each of the tests in Problem 1 that are executed successfully. (Note that the check_list routine in qsort_hypercube.h needs to be modified to verify descending order.)**

   Attached code snippet *qsort_hypercube_descending.c* and *qsort_hypercube_descending.h*