Assignment 2 Report
Akshay Satam (111481679), Prateek Roy (111481907)

README:

Question 1:

Question 1 has two files - 1 file for each sorting algorithm (quick sort and radix sort).

In each file, I have written one function for each subquestion. To run any subquestion, please comment the other functions and run the file.

Note:
When testing radix sort (q2.cpp), please be careful to change the variable “p” (which corresponds to the number of processors) in the functions - “par_radix_sort_with_counting_rank” and “par_counting_rank” to the desired value. Also, we are setting the number of processors programmatically for each subquestion (i.e. in each subquestion corresponding to the subquestion). So please change that as well.

Question 2:

Question2 has two files for MST with radix+counting sort and MST with Binary Search.
Just do make and run by :

1) make
2) ./run < path_of_input_file

The output files of all runs are inside output folder.

Please find the answers to the questions:

1b.
For the first part of this question, I kept on increasing the input size until the average execution time took less than 15 seconds. After taking average of three runs, I found the value of **N** to be **2^21 = 2097152.**
For the above executions, I kept the value of base case (m) to be 32. Then, for the above value of N (2^21), I kept on increasing the value of “m” to find the optimal value of “m”. I found that the execution time kept on reducing by increasing “m” until “m” became 2^21 (i.e.2097152). The decrease in execution time is not linear with decrease in M. After this value, the time seemed to be more or less constant. Please find the table, showing value of m versus the execution time.

Hence, the value of M is **2^21 (i.e. 2097152)**

M	**Time (sec)**
32	15.75
64	13.68
128	12.48
256	10.62
512	9.42
1024	6.89
2048	5.67
4096	3.67
8192	3.31
16384	2.48
32768	2.32
65536	2.10
131072	1.41
262144	1.24
524288	0.99
1048576	0.89
2097152	0.086

Question 1c:
When using one processing element, I found that Quicksort takes less than two minutes for input size of **2^27**. Above this value, quicksort takes more than 2 minutes. Hence, the maximum size of input for which the execution time of both sorting algorithms does not exceed 2 minutes is **2^27**.

Please find the values of the two sorting algorithms for input size **2^27**.

Part 1

Algorithm	**Time (sec)**	**R**
Quicksort	90	27
Radix sort	20	27

Part 2

We kept on increasing the processors and found the execution time as follows.

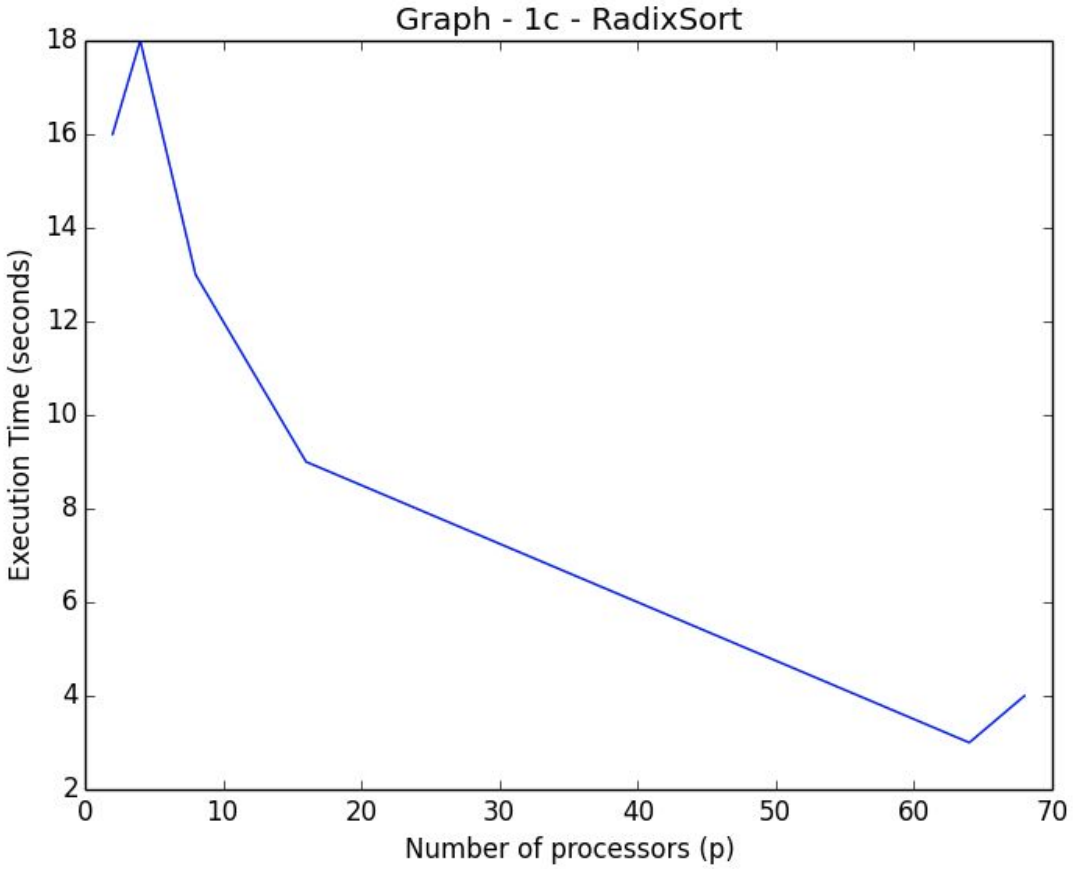
The general trend we observed is that the execution time reduced by increasing the number of processors. However, in case of Radix sort, there were two exceptions.

1. The execution time is more for 4 processors as compared to the execution time of 2 processors.
2. The execution time is more for 68 processors as compared to the execution time of 64 processors.

All other cases behave as expected.

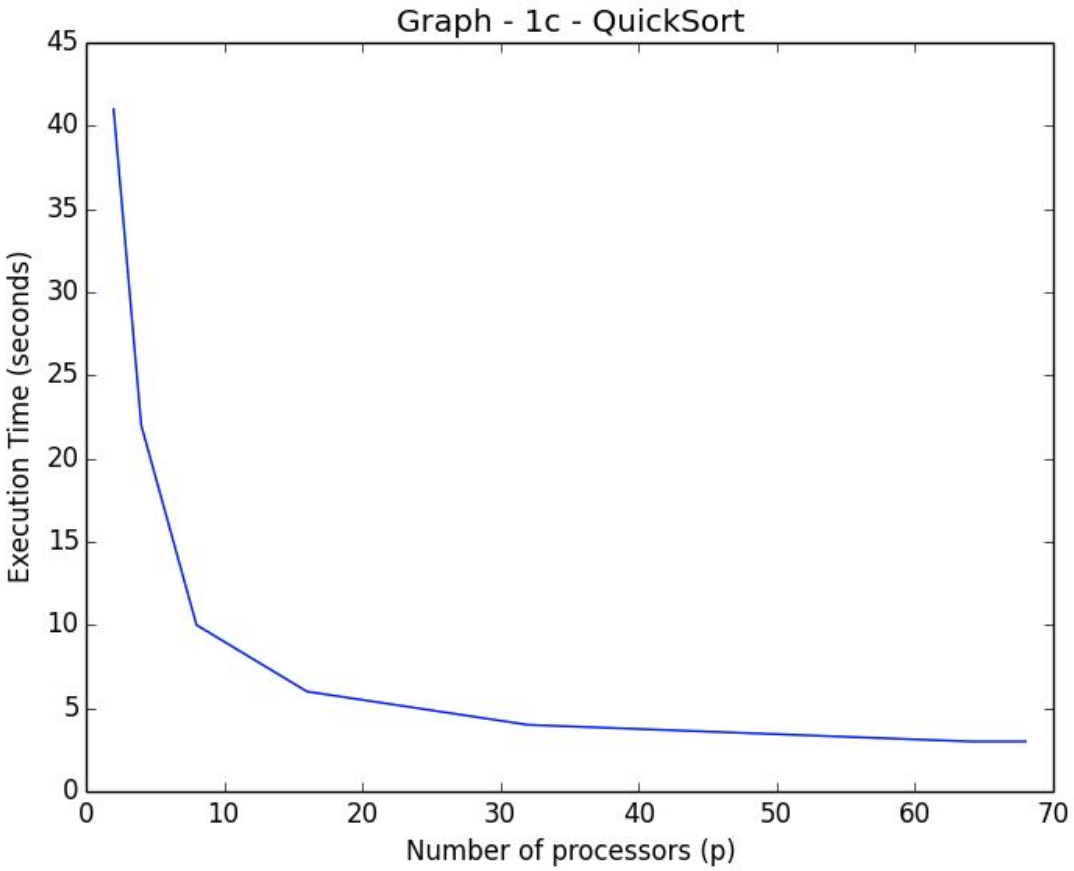
Radix sort

No of Processors	**Time (sec)**
2	16
4	18
8	13
16	9
32	7
64	3
68	4



Quicksort:

No of Processors	**Time (sec)**
2	41
4	22
8	10
16	6
32	4
64	3
68	3

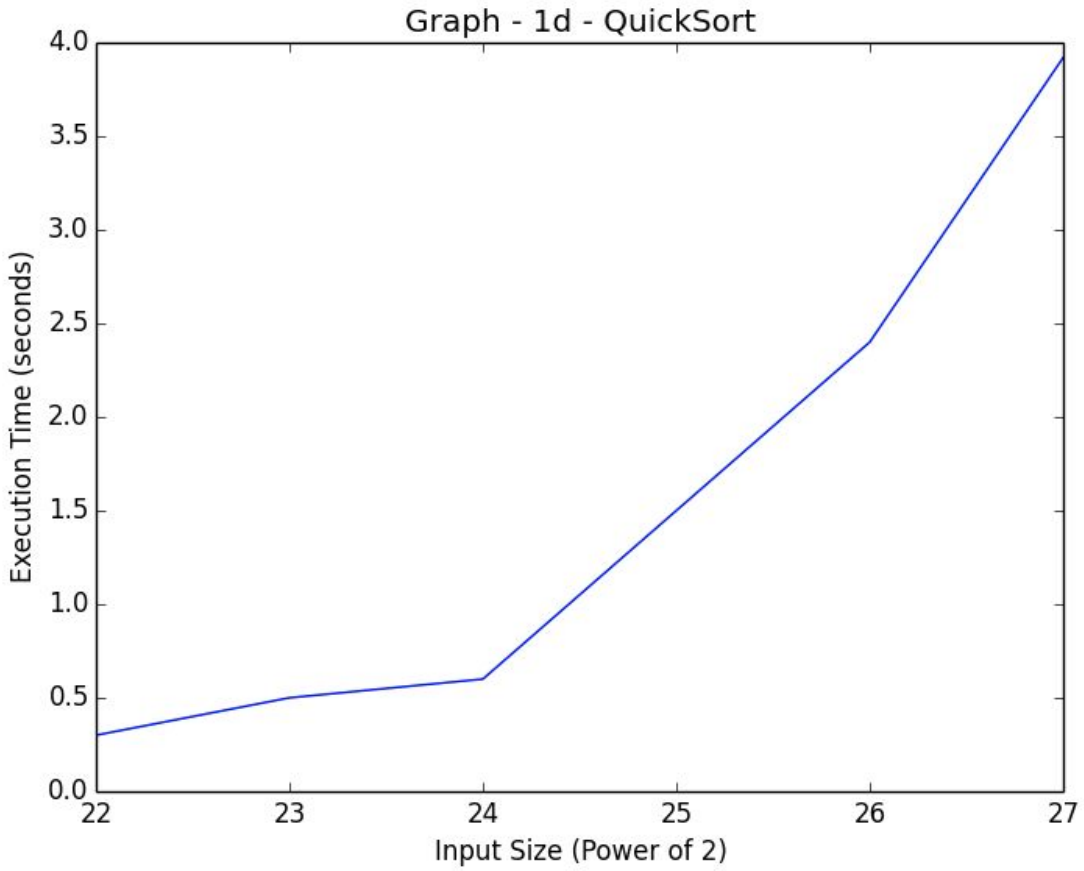


Question 1d:

For this question, we used all the processors and increased the size of the input. We found that the execution time increases with increasing the input size. The increase in not linear though.

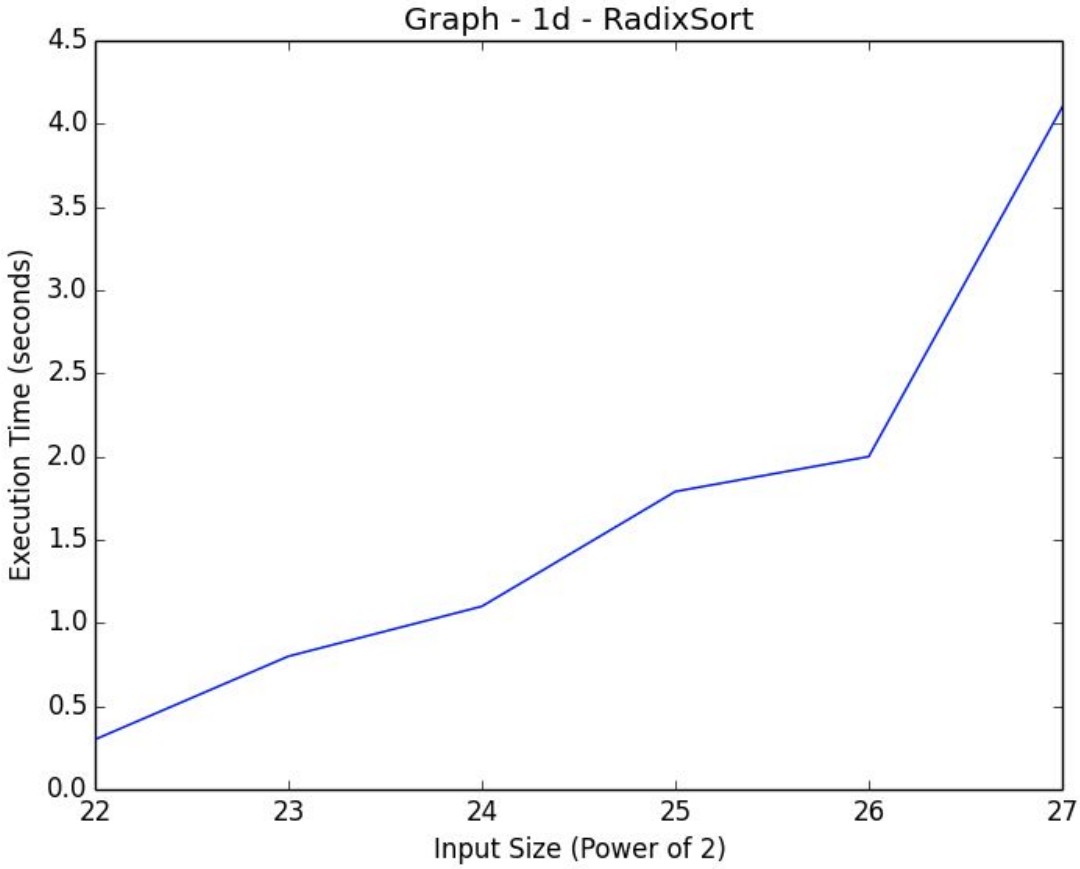
Quicksort:

N (power of 2)	**Time (sec)**
22	0.3
23	0.5
24	0.6
25	1.5
26	2.4
27	3.92



Radixsort:

N (power of 2)	**Time (sec)**
22	0.3
23	0.8
24	1.1
25	1.79
26	2
27	4.1

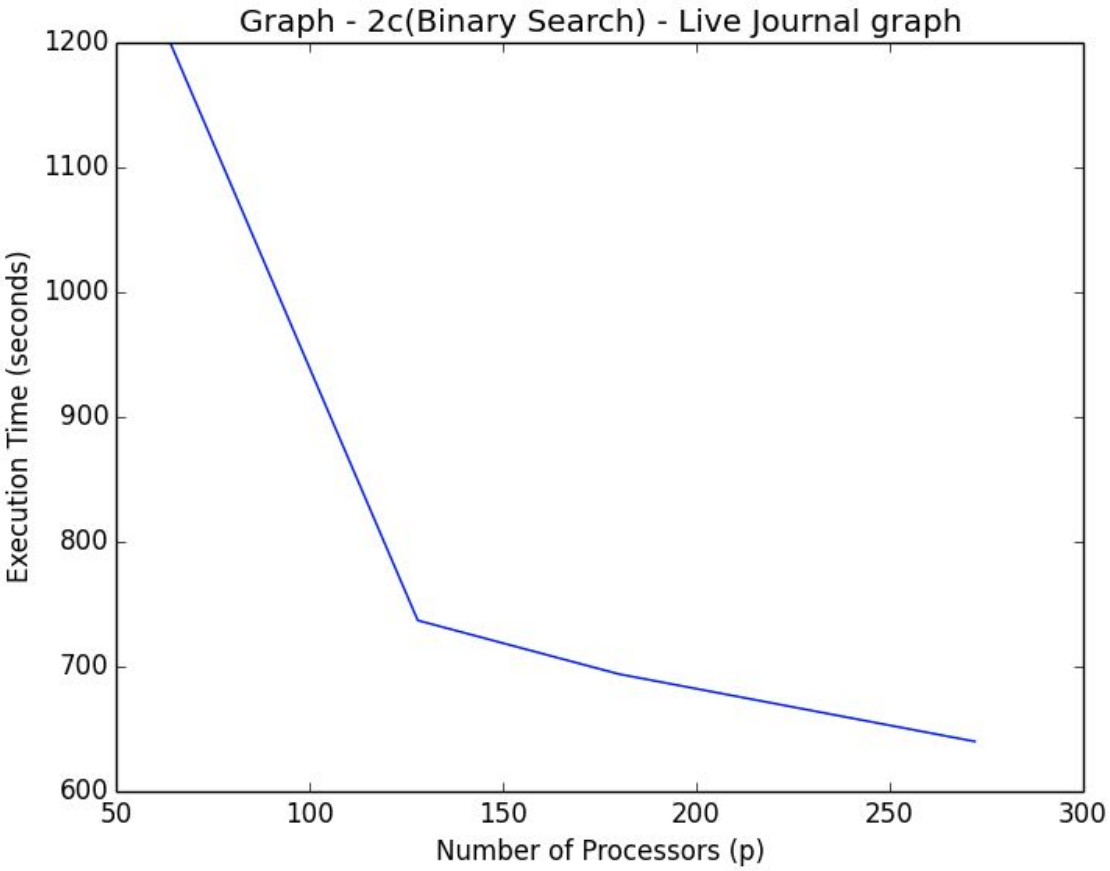

Graph - 1d - RadixSort
Execution Time (seconds)
Input Size (Power of 2)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
22
23
24
25
26
27

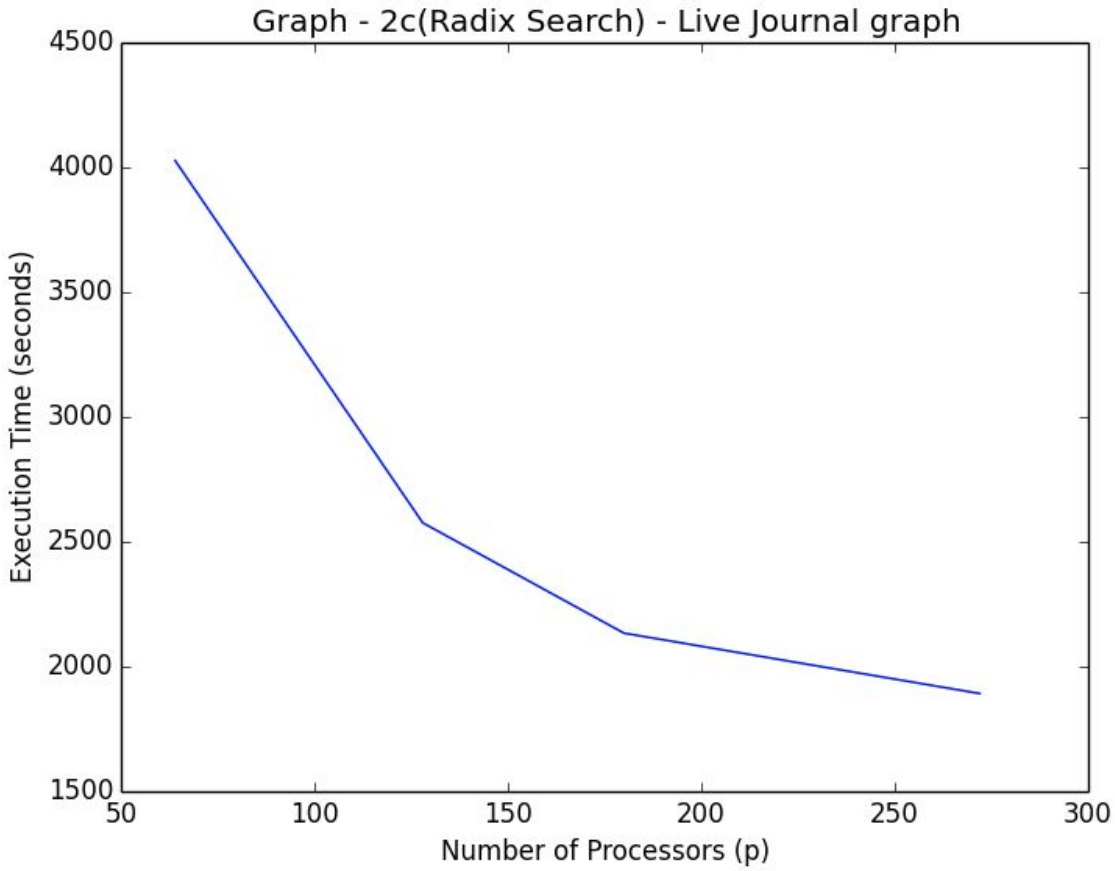
Q2.

Cores	RadixSortWithRanking (seconds)	BinarySearch (seconds)
ca-AstroPh-in.txt	42	5
com-amazon-in.txt	120	25
com-dblp-in.txt	124	24
roadNet-CA.txt	383	117
roadNet-PA	279	24
roadNet-TX	311	38
as-skitter	1187	199
com-lj	3467	640
com-orkut	Timeout	Timeout
com-friendster	Timeout	Timeout

2c) Live Journal Graph

Cores	BinarySearch (seconds)	RadixSortWithRanking (seconds)
64	1200	TimeOut
128	737	2576
180	694	2134
272	640	1892


Graph - 2c(Binary Search) - Live Journal graph
Execution Time (seconds)
Number of Processors (p)
1200
1100
1000
900
800
700
600
50
100
150
200
250
300



From both the above graphs we can see that as the number of processors increase, execution time decreases. So it is **Strongly Scalable**.