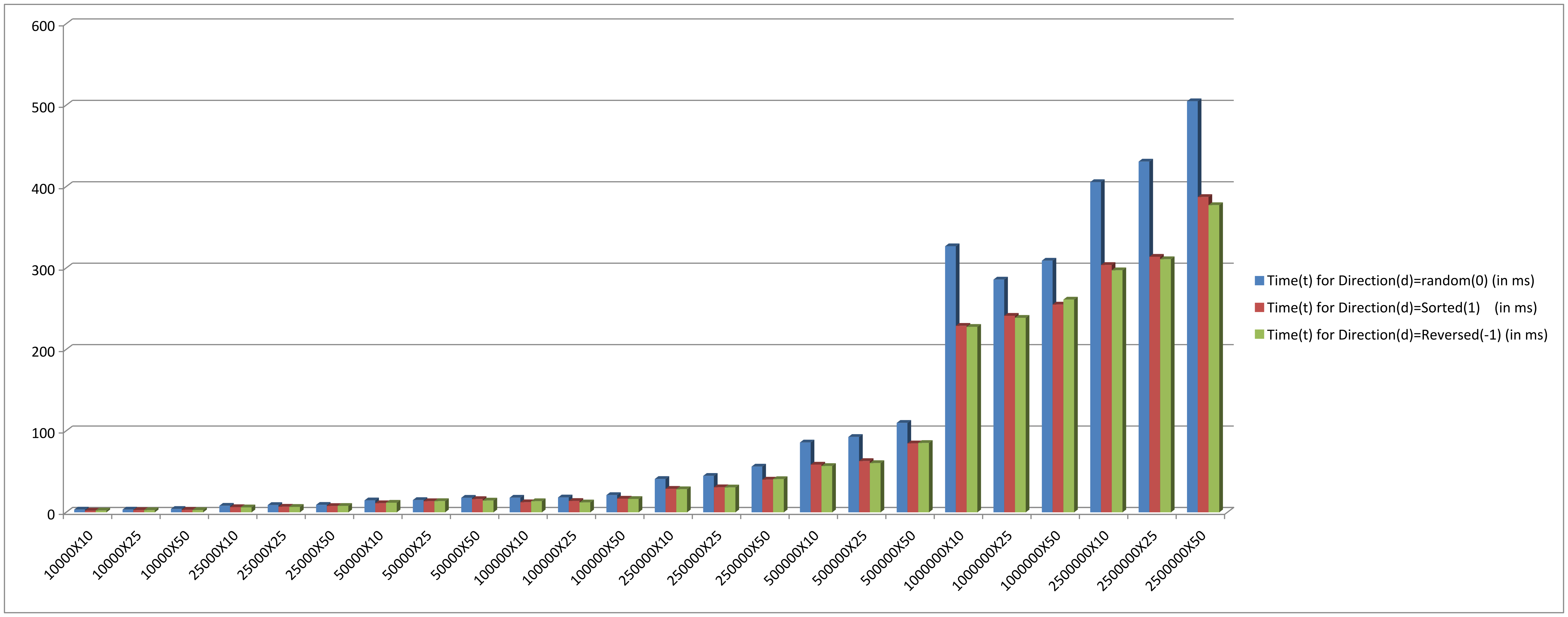
Merge Sort Run Time Report :

Input Size(m) X dimenstion(n)	Time(t) for Direction(d)=random(0) (in ms)	Time(t) for Direction(d)=Sorted(1) (in ms)	Time(t) for Direction(d)=Reversed(-1) (in ms)
10000X10	3.4	2.1	2.3
10000X25	3.4	2.8	2.7
10000X50	4.2	3.1	2.8
25000X10	8.1	6.4	6.1
25000X25	9	6.9	6.7
25000X50	9.2	7.8	7.9
50000X10	14.7	11.2	11.8
50000X25	15.1	13.7	13.8
50000X50	17.9	16.3	14.5
100000X10	18	12.4	13.7
100000X25	18.3	14	12.2
100000X50	21.2	16.8	16.4
250000X10	41	28.9	28.4
250000X25	44.8	30.8	30.6
250000X50	56.2	40.2	40.8
500000X10	85.8	58.6	56.9
500000X25	92.6	63	60.5
500000X50	109.9	84.7	85.2
1000000X10	326.8	229.1	227.8
1000000X25	285.9	241.4	238.9
1000000X50	309.1	255.2	261.3
2500000X10	405.6	303.8	297.3
2500000X25	430.8	313.9	310.9
2500000X50	504.9	387.3	377.3



<u>Analyis of Merge Sort Algorithm with the inputs provided:</u>

1) Merge Sort Algorithm's execution time is directly proportional to the input size. Run time increases as the size of the input grows.
2) Implementation of merge sort works in O(nlogn) times in all 3 cases - (Best,Average and Worst).
3) Therefore, there is a minute difference between execution time when input is sorted(best case) and reversed(worst case).
4) However, in some cases when the input direction(d) value was -1(reversed), observed that the algorithm's execution time was less than that of the case when the inputs were sorted.
5) Readings are taken for merge sort function alone : ie other functions such as insertion_sort() and insertion_sort_im() are commented out in main.cc while taking readings for merge sort.