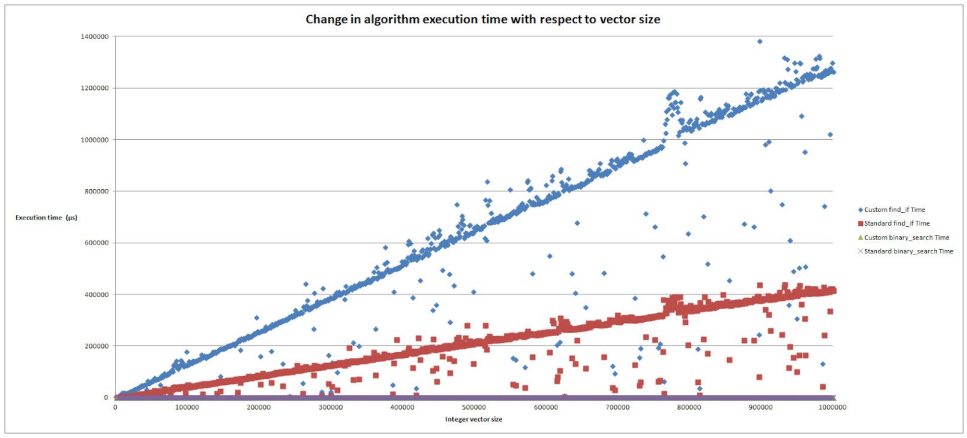
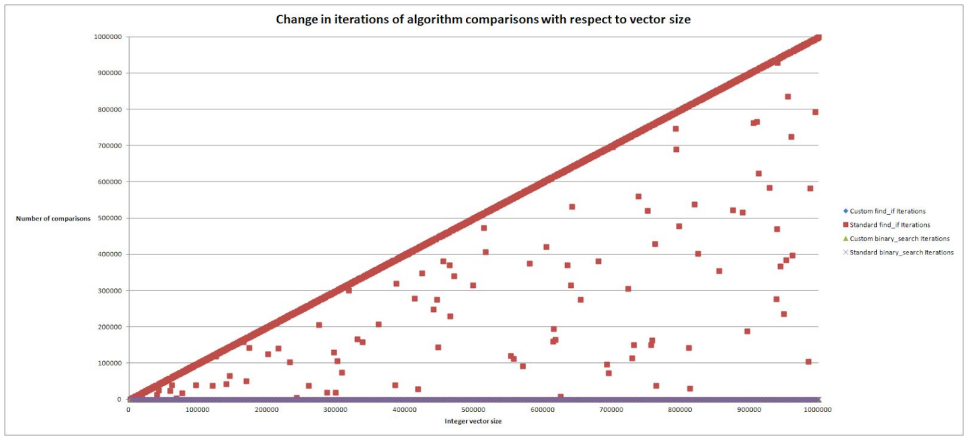


As can be seen in the above graph, the execution time of both the custom and standard find_if algorithms were orders of magnitude greater than those of the custom and standard binary_search algorithms. In fact, this difference is so great that it is the reason why the execution time was measured in microseconds. Any larger subdivisions (for example, milliseconds) would lead to the execution time of both binary_search algorithms to be rounded down to zero. It is unclear why the custom find_if algorithm time grew at a faster rate than the standard find_if time; should further analysis be conducted, this would be an interesting question to explore.



The custom and standard versions of the find_if algorithms have an almost perfect correlation between vector size and number of comparisons. The custom and standard versions of the binary_search algorithms, however, never exceed 21 comparisons. Thus, it appears that the binary_search algorithms are much more efficient than the find_if algorithms in terms of the number of comparisons made. Both of these graphs indicate that the binary_search algorithms are several times more efficient than their find_if counterparts. This data, however, does not take into consideration the amount of time it takes to sort the vector before the beginning of the binary_search algorithms. Since they do not work if the given list is not sorted, it would not be unreasonable to consider this sorting a part of the algorithm's execution time.