a. Hypothesis

I expect a sorted vector to have pretty much the same times as a multiset in C++. Binary Search takes log(n) time so I would think that insertion with binary search would be log(n) time as well. I believe that multiset insertion is also log(n) time so that is why I would expect very similar times. I am using C++ to implement the sorted vector so I do not think that there will be any difference from processing time or compile time because I used C++ to measure multiset times. There might be a small difference considering I have to write my own class and there will be extra calls, but I expect the time difference to be minimal if not indistinguishable.

b. Methods

I decided to use Visual Studio with the Microsoft C++ Compiler so I could easily compare my results to those from my previous experiment. This time I decided to use the default optimization because using the O2 Max Speed optimization did not have quite the same effect on a sorted vector than it did on a multiset. To implement a sorted vector in C++ I created a class containing a private vector variable and a public method to insert. The insertion method used a binary search to find the insertion point then inserted the given number at that point. I used the same range of N values from my previous experiment (10,100,1000,10000,100000 … 900000). I looped and incremented N until it reached 900,000 and each repetition, I created N random numbers and timed how long each insertion to the sorted vector took. I then printed this data to compare against the multiset.

c. Results

Shown below (figure 1) are the results from the experiment with an Ordered Vector and Multiset in C++. The points represent the insertion time taken in seconds (y-axis) for a value of N (x-axis). The N values range from 10-900,000 and the times from 0-24 seconds. The orange line represents a plot of the sorted vector insertion times and the blue line represents a plot of the multiset insertion times.

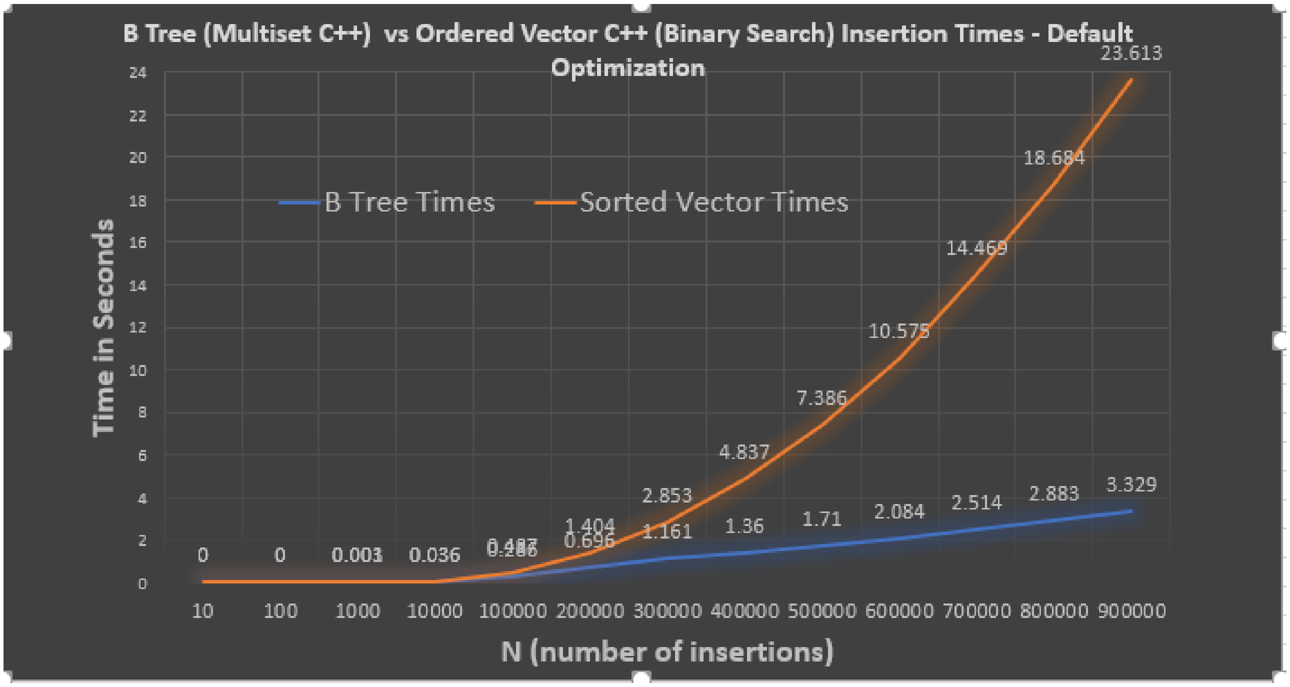

Figure 1

d. Discussion

The data found was surprising and not what I had expected to get. The insertion times of the sorted vector were much higher than the multiset and do not seem to be O(log(n)) time. I am surprised because I am almost positive that my method to do binary search and insertion is efficient and log(n) so that makes me even more surprised that I would get such higher times. I would expect that my own class and method might take a little longer because STL methods are made as efficient as possible and written by professionals, but I did not think that it would make this much of a difference. It is possible that my algorithm for binary search is not optimized but I believe it follows the binary search algorithm very closely. I expect that the sorted vector took more time because it first had to binary search for the insertion point then shift the vector values once inserted taking extra time.

e. Conclusion

Under 100,000 insertions the difference between std::multiset and my sorted vector are indistinguishable but for insertions greater than 100,000 std::multiset is faster than the sorted vector using binary search insertion.