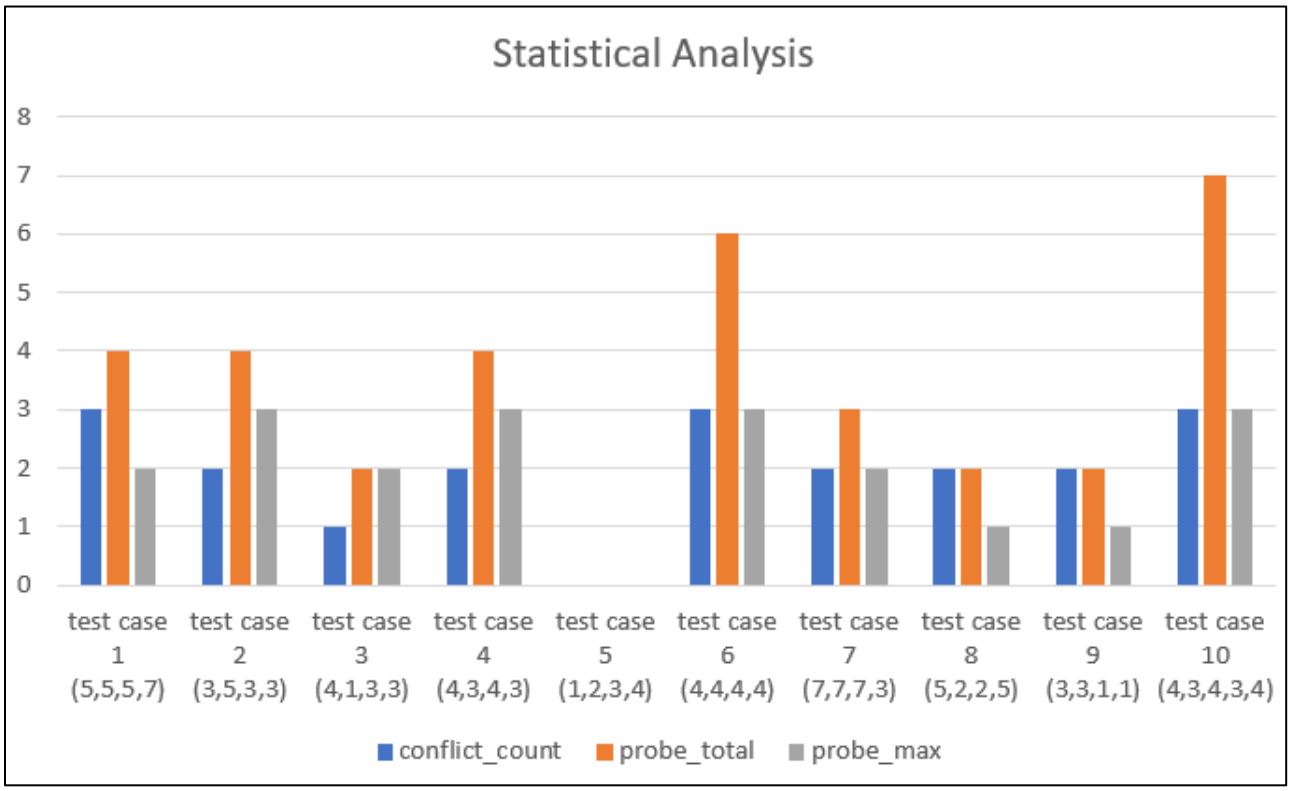
Statistical Analysis of the Hash Table Performance

Method Returns

- The total number of conflicts (conflict_count)
- The total distance probed throughout the execution of the code (probe_total)
- The length of the longest probe chain throughout the execution of the code (probe_max)

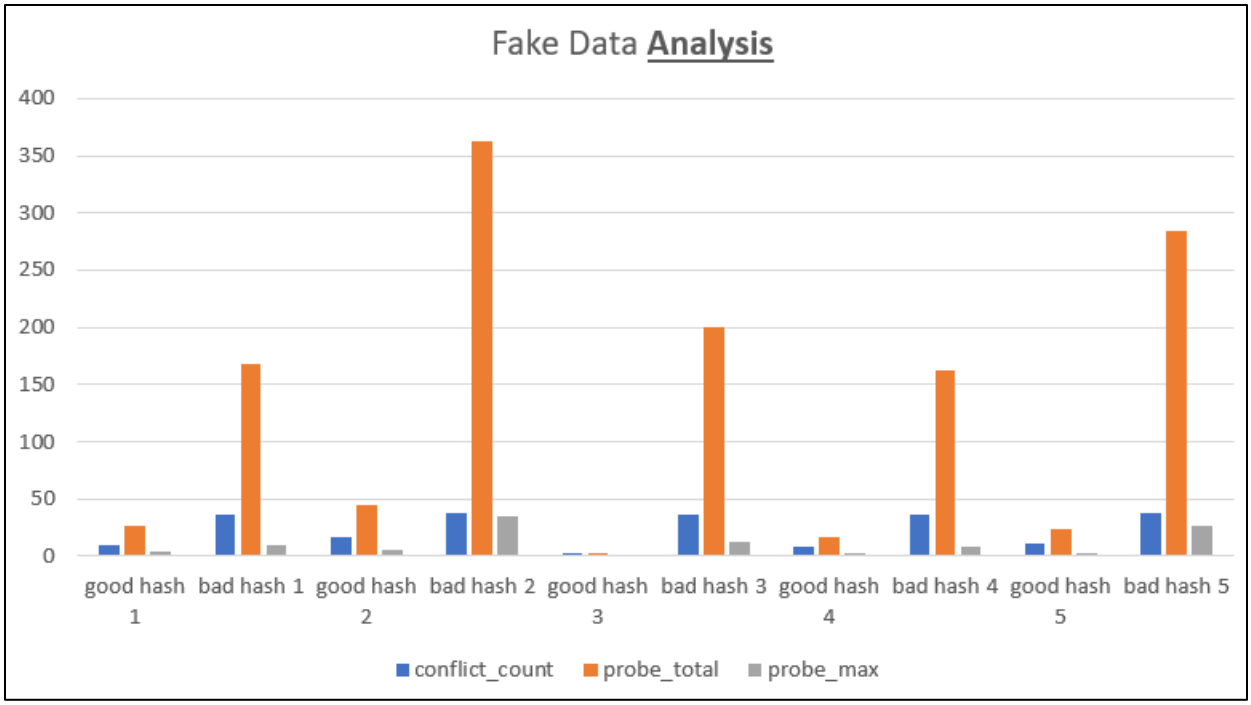
By providing fake data on to test cases we analyze the piece of code. From these cases plotting them graphically we get



we can first assume that the total distance probed is always higher than the other returns, furthermore we also conclude that if all the values have no similar keys there are no conflicts which would return no probe and thus returning 0,0,0.

As we can see from the above graph both our assumptions are right. On further analysis of the graph we see that the order of the conflict **(h(sn)=h(sm))** matters in determining the length of the longest probe chain, for example if the conflict happened between the first 3 inputs our probe_max would return 2 whereas had it occurred in the last 2 inputs it would have returned 3 since that location is already being used and the next one is also occupied.

We can also see that as the elements increase, we have the total distance probed also increases but since we are using the good hash the length of the longest probe chain is always as min as possible (here under 3).



Here under the Fake Data Analysis, we can see all the return values for the good hash function is always less than the bad hash functions leading to our good hash function being more efficient compared to the bad hash function. Due to the good hash function distributing the keys uniformly hence minimizing the number of collisions , by graph we can see that the bad hash doesn't do well in this part in turn resulting in a higher total distance probed during the execution in every situation.