1. Understanding Hashing & Collision Resolution Technique

   https://www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/

   https://www.sparknotes.com/cs/searching/hashtables/section1/

2. Characteristics of a Good hash Function & Mostly used Hash Function

   https://www.sparknotes.com/cs/searching/hashtables/section2/

3. Implementing a Hash Table from Scratch

   https://www.sparknotes.com/cs/searching/hashtables/section3/

   a. Use a good hash Function that Uniformly distribute the keys across the possible range of hash value, i.e we need hash value to be uniformly in range [a,b]
   b. Even If we choose a good hash Function, there are chances of collision
   c. Use Separate chaining to resolve collision (For every index, we store a linked list to store every element hashed to same index)

   Even if the input data Strings are almost identical (very less change in the characters or any permutation of a single string), the hash Function takes care of uniformly Distributing the keys across hash table.

But somehow if there are collisions or every input data is same , the load factor for any index in hash table can increase.It results in worst case time Complexity of Lookup is O(n).

But , Best case and Average case complexity is O(1).

We can use a Balanced BST kind of structure for separate chaining (Collision Resolution) to Assures the Worst case Lookup is also done in O(log n) .

   d. We can keep on changing the Hash_Table Size (start from 32) and increase the size as the Load increases for better memory performance.It also helps in reducing collision .

https://www.sparknotes.com/cs/searching/hashtables/section3/

http://javabypatel.blogspot.com/2015/10/what-is-load-factor-and-rehashing-in-hashmap.html

https://www.geeksforgeeks.org/unordered_map-load_factor-in-c-stl/
http://www.cplusplus.com/reference/unordered_map/unordered_map/load_factor/

4. Understanding Unordered_map in C++ (How to use User Defined class as keys)

https://codeforces.com/blog/entry/21853

http://www.cplusplus.com/reference/unordered_map/unordered_map/

https://www.geeksforgeeks.org/unordered_map-in-cpp-stl/