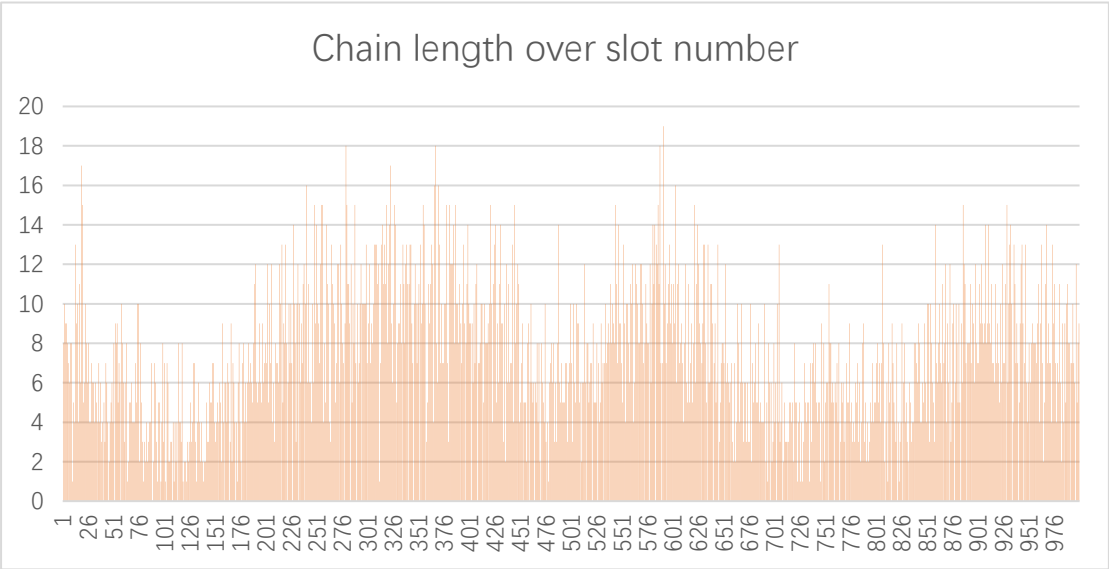
The required bar graph is demonstrated as below:



The minimum number of words assigned to a slot is 0, and the maximum number is 19. The standard deviation is 3.4.

My hash function is implemented based on the following considerations: Each character of a string can be automatically converted into an integer in the C++ language, so the trick is to find a way to compose all the characters in the string. A simple rough idea is to accumulate all the characters one by one, but it comes with an obvious disadvantage: Words that share the exactly same alphabet will have the same hash value. For example, the word “abcde” will have the same hash value as the word of “edcba”. This is not good as it will lead to a huge number of collisions. My solution is intuitive: Not only the character itself matters when hashing, the sequence of its occurrence matters as well. Still taking the word “abcde” as example, its hash value shall be calculated as:

$$a*1 + b*2 + c*3 + d*4 + e*5$$

And it is obvious that the value should be different from the hash value of “edcba”.

Based on the statistics presented above, the resulting chain length varies in the range of [0, 19] with a standard deviation of 3.4, which is acceptable for me.