Project Report

GONG Xi-57125169

I first choose the Division method(P1) to construct a hash function. And used the Linked List to handle the collision. If the key value collides, I just add the value to the end of the list.

图形用户界面

中度可信度描述已自动生成

P1: Division method

形状

中度可信度描述已自动生成

P2: Linked List

Second, I choose the Mid-square method(P3).

文本

描述已自动生成

P3: Mid-square method

For the third one, I choose to multiply the key with a Fibonacci number to produce the new key. Since the Fibonacci sequence is a magical sequence, multiplication may help to distribute the key better.

文本

描述已自动生成

P4: Fibonacci method

To summarize a better conclusion. I chose to keep the solution of the collision so that I could better understand how the hash function may influence the result.

The result of my testing

I tested my three algorithms with five groups of data：

图形用户界面, 文本, 应用程序

描述已自动生成

Method: Division Method



Seed1: 114514 Seed2: 57125169 Seed3: 3334 Seed4:33342 Seed5: 13354

Method: Fibonacci method

    

Seed1: 114514 Seed2: 57125169 Seed3: 3334 Seed4:33342 Seed5: 13354

Method: Mid-square Method

    

Seed1: 114514 Seed2: 57125169 Seed3: 3334 Seed4:33342 Seed5: 13354

To my surprise, except for the first two groups of method cost nearly the same time, the third method, the Mid-square method cost nearly four times as long as the former two methods. In my opinion, it should be related to the complexity of the hash function since every time you do a hash operation, and you have to go through the hash function once. So, from the tests, I know that when I design a hash table, I should not make the hash function too complicated.

I guess the Mid-square method would not be one of my choices when I write a hash function afterward.

After the last test. I tried to enlarge the size of my hash table simply tries to find out how it could help me. In the first attempt (the bottom one), the size is 1024, and the following becomes 10244 and 102444. We could see that at first, the large memory indeed helps to save time. But maybe when the memory becomes larger and larger, most of the space is useless. And even if the input is small, it also needs such a large space. So, I guess maybe it is better to enlarge space when it is necessary. However, we could still find out that when the input data is larger, the running time will be longer.

