1 布谷鸟哈希程序设计

参考了课本的代码，并且在其基础上进行了简化。首先，概述这里的布谷鸟哈希 insert 实现流程（与上课讲的双表型有所不同）：

1. 初始化一个哈希表，长度可以设置（默认大小 101），其中所有位置都是空的。备用哈希函数的个数可以设置（默认 3 种）
2. 当用 insert 方法插入元素时，会根据元素的类型去调用对应的哈希函数（我这里有整型和字符串两种），根据计算得到的位置尝试存入
3. 如果该位置已经有元素占用了，使用下一个备选哈希函数……若存在一个备用哈希函数，使得该元素经其映射得到的位置是空的，则将该哈希函数填入之，返回。否则，转入下一步
4. 踢除操作：任选一个备选哈希函数，该元素强行进入经其映射到的那个位置里，并把原来的老元素踢出来，再对被踢出来的老元素使用 insert 方法，如此循环。
5. 如果发生踢除操作的次数大于阈值，则认为表太小，调用 rehash 方法扩增，更换新的哈希函数，再把表的大小翻为原大小的两倍以上。然后，把原来的元素（包括刚才填不进去的那个元素），全部 insert 一遍到新表里

代码说明：

1. 一共就只有一张表（这样的实现一般来说会比双表更省空间）

2. 与课本代码不同，我简化了 rehash 的判别，如果在某一次 insert 之中，造成的碰撞次数超过阈值（阈值我设为当前元素总数的四分之一，和 100 取 min 的值）就 rehash（无需像课本代码有个内置的变量去记录累计量）

3. 与课本代码不同，调整了哈希函数的生成规则

4. 增加了打印哈希映射、哈希表的 public 函数，便于可视化结果与 debug

2 测试思路

2.1 基本功能正确与否

测试两种类型（字符串和整型）分别作为基本元素时，哈希表是否工作正常，首先是两个含有可视化的函数，会显示出此次的哈希函数是哪些，也会在每一步都打印出此时的表的大小、此时存有的元素的个数，也会打出整张表。此外，也会在触发布谷鸟踢除时，显示出 “kick” 了哪个元素，在触发 rehash 时会打印 “rehash”，方便检查。也测试了 remove 函数以及 contain 函数的正确实现。

2.2 效率考察

在测试时使用 chrono 进行计时，前后的时间作差。效率测试，就关闭 debug 模式（不会输出每一步的步骤）。此外，对于每一次测试时，输入数据的长度都比表格的**初始大小**的一半小，这样保证了装载率小于 0.5。而且，我设置填入数据的范围也和初始表格的大小成正比，这样的话，由于数据重复而填入失败的比例将会接近。

3 测试效果

可直接使用 make run 命令编译且运行。

输出形如：

```
****************************************
************* TEST STRING **************
****************************************

Multipliers: 47 53 59
        hello
        hello                   APQkj
Table Size is 5
current elemment size is 2
        hello           LFaTy   APQkj
Table Size is 5
current elemment size is 3
        hello   nrXMU   LFaTy   APQkj
Table Size is 5
current elemment size is 4
YJQNf   hello   nrXMU   LFaTy   APQkj
Table Size is 5
current elemment size is 5
<<<<<<<<<<< kick!!!<<< hello <<<<<
<<<<<<<<<<< kick!!!<<< APQkj <<<<<
############## rehash trigger ##############
LFaTy   ypoZR   APQkj   nrXMU   hello                                   YJQNf
Table Size is 11
current elemment size is 6
LFaTy   ypoZR   APQkj   nrXMU   hello                           WVpxm   YJQNf
Table Size is 11
current elemment size is 7
LFaTy   ypoZR   APQkj   nrXMU   hello                           WVpxm   YJQNf   fHvsm
Table Size is 11
current elemment size is 8
LFaTy   ypoZR   APQkj   nrXMU   hello   vOnkf                   WVpxm   YJQNf   fHvsm
Table Size is 11
current elemment size is 9
LFaTy   ypoZR   APQkj   nrXMU   hello   vOnkf           MzZKi   WVpxm   YJQNf   fHvsm
Table Size is 11
current elemment size is 10
<<<<<<<<<<< kick!!!<<< fHvsm <<<<<
<<<<<<<<<<< kick!!!<<< ypoZR <<<<<
LFaTy   fHvsm   APQkj   nrXMU   hello   vOnkf   ypoZR   MzZKi   WVpxm   YJQNf   rxorM
Table Size is 11
current elemment size is 11
contain hello? 1
LFaTy   fHvsm   APQkj   nrXMU           vOnkf   ypoZR   MzZKi   WVpxm   YJQNf   rxorM
current elemment size is 10
```

```
contain hello? 0
same is 0
****************************************
*************** TEST INT ***************
****************************************

Multipliers: 457 509 563

        104133
        104133          39294
Table Size is 5
current elemment size is 2
        104133  31681   39294
Table Size is 5
current elemment size is 3
13690   104133  31681   39294
Table Size is 5
current elemment size is 4
13690   104133  31681   39294   58338
Table Size is 5
current elemment size is 5
<<<<<<<<<<<< kick!!!<<< 58338 <<<<<
<<<<<<<<<<<< kick!!!<<< 45813 <<<<<
############## rehash trigger ##############
        104133                  13690   39294   45813   58338   31681
Table Size is 11
current elemment size is 6
31493   104133                  13690   39294   45813   58338   31681
Table Size is 11
current elemment size is 7
31493   104133                  13690   39294   45813   58338   31681           67324
Table Size is 11
current elemment size is 8
31493   104133          81245   13690   39294   45813   58338   31681           67324
Table Size is 11
current elemment size is 9
<<<<<<<<<<<< kick!!!<<< 58338 <<<<<
<<<<<<<<<<<< kick!!!<<< 67324 <<<<<
<<<<<<<<<<<< kick!!!<<< 58338 <<<<<
############## rehash trigger ##############
<<<<<<<<<<<< kick!!!<<< 39294 <<<<<
<<<<<<<<<<<< kick!!!<<< 104133 <<<<<
        104133                          67324                           58338   31493
Table Size is 23
```

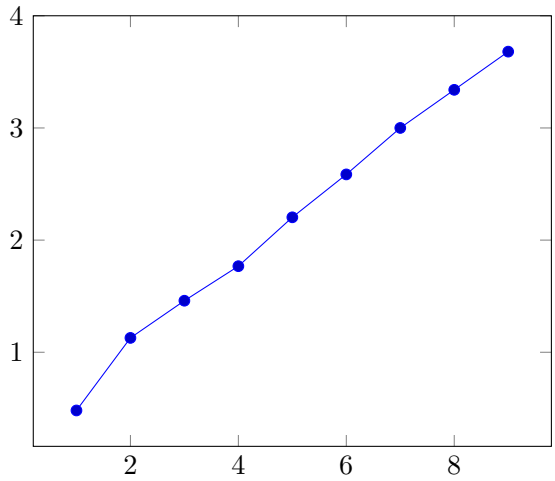
```
current elemment size is 10
        104133          23103           67324                           58338   31493
Table Size is 23
current elemment size is 11
contain 104133? 1
                        23103           67324                           58338   31493
current elemment size is 10
contain 104133? 0
same is 0
****************************************
************ TEST EFFICIENCY ************
****************************************

Table Size is 10000000
Table Size is 10000000
current elemment size is 4974826
same is 25174
TIME COST 0.481045

Table Size is 20000000
Table Size is 20000000
current elemment size is 9948910
same is 51090
TIME COST 1.12801

Table Size is 30000000
Table Size is 30000000
current elemment size is 14916206
same is 83794
TIME COST 1.45952

Table Size is 40000000
Table Size is 40000000
current elemment size is 19894969
same is 105031
TIME COST 1.7675

Table Size is 50000000
Table Size is 50000000
current elemment size is 24807594
same is 192406
TIME COST 2.20335

Table Size is 60000000
```

```
Table Size is 60000000
current elemment size is 29627171
same is 372829
TIME COST 2.58583

Table Size is 70000000
Table Size is 70000000
current elemment size is 34218790
same is 781210
TIME COST 3.00016

Table Size is 80000000
Table Size is 80000000
current elemment size is 37396700
same is 2603300
TIME COST 3.33996

Table Size is 90000000
Table Size is 90000000
current elemment size is 40393100
same is 4606900
TIME COST 3.68048
```

上述结果已经加入-O2 优化，如下（单位：second）：

表 1: 运行时间对比

序号	**大小**	**运行时间 (second)**
1	10000000	0.481045
2	20000000	1.12801
3	30000000	1.45952
4	40000000	1.7675
5	10000000	2.20335
6	20000000	2.58583
7	30000000	3.00016
8	40000000	3.33996
9	90000000	3.68048



用 valgrind –leak-check=full ./test 进行测试，发现没有发生内存泄露

4 时间复杂度分析

根据上面的测试，仅仅从运行耗费时间的角度，能够验证总的时间复杂度是 $\Theta(n)$ 这一点（即每一步都是 $O(1)$ 的），基本上每增加 10000000 个插入的数据，就会增加 0.4 秒左右的耗时。