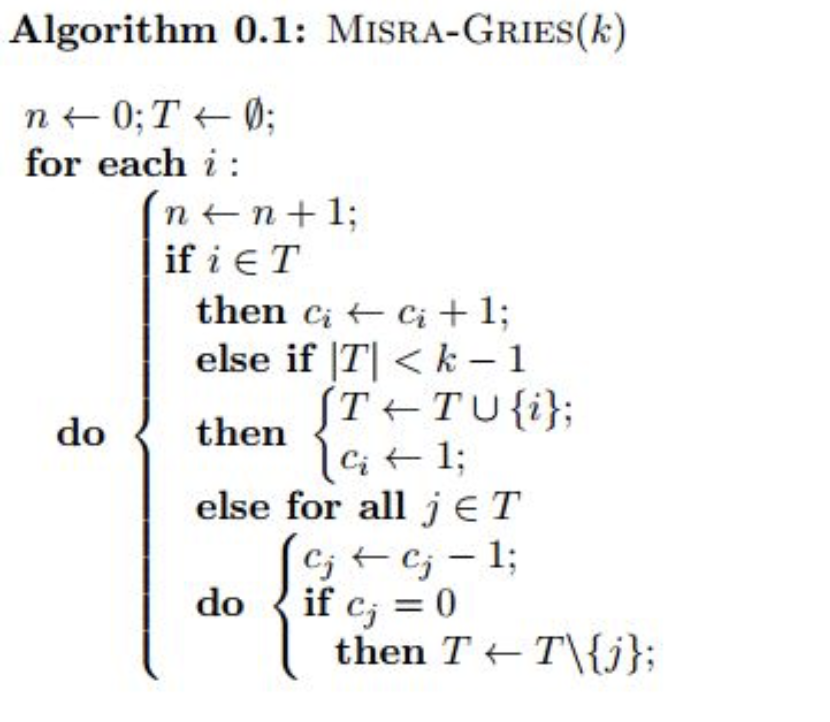
**第8周上机实验报告**

# 一、题目：

实现Misra-Gries algorithm

****

# 二、算法思路：

## MG算法

（1）处理一个新到来的元素x时

（2）If已经为其分配计数器，增加之

（3）Else If没有相应的计数器，但计数器个数少于k，为x分配计数器k，并设为1

（4）Else所有的计数器值减1，删除值为0的计数器

## 加快MG算法处理单个元素的效率

设计一个增、删、查、改均时间复杂度较低的数据结构

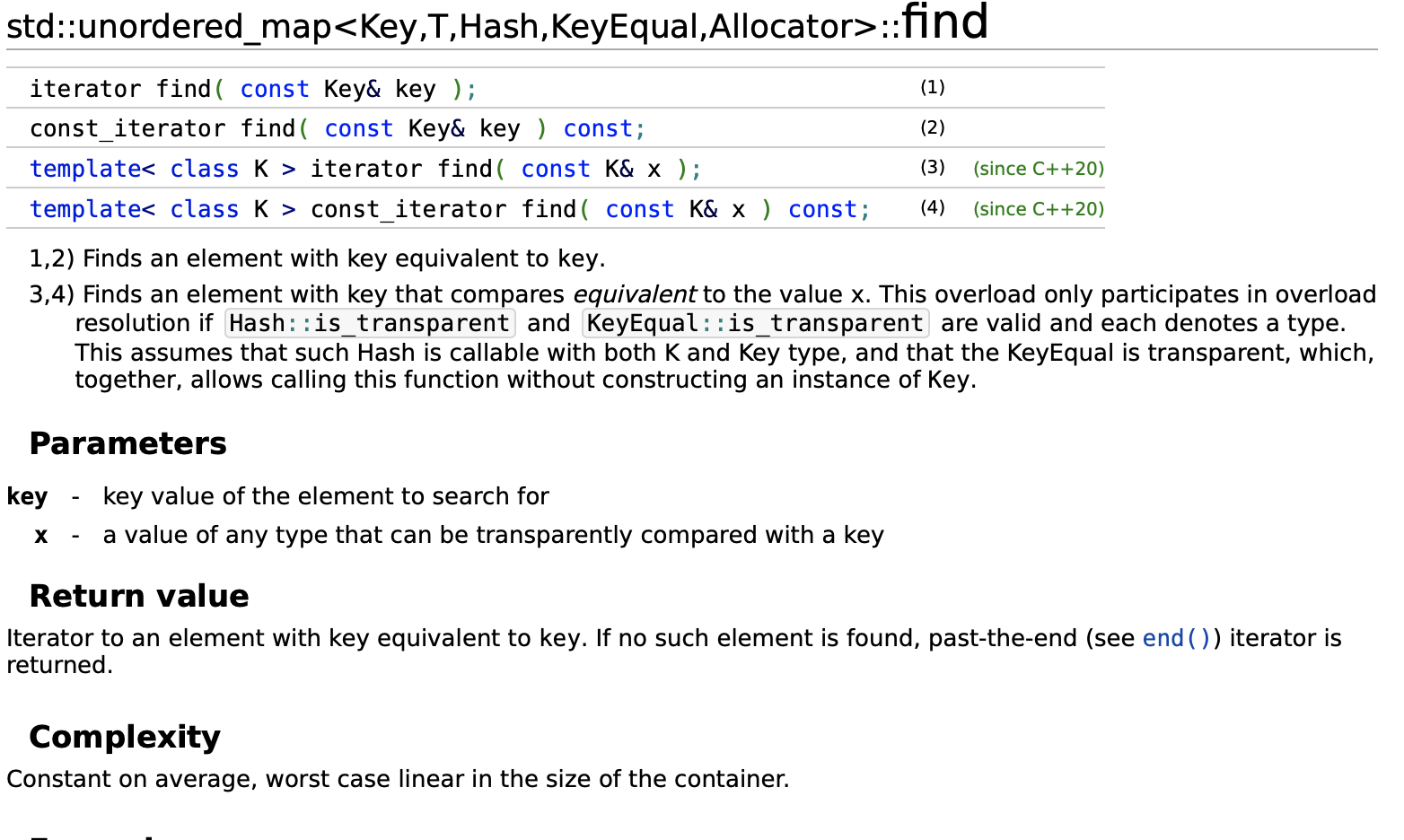
### 哈希表和动态数组的组合

* 哈希表
  + 存储元素i和其在动态数组中位置的映射
* 动态数组
  + 储存一个pair<int,int>
    - First：元素i
    - Second：元素i的出现次数

1. **private**:
2. unordered\_map<**int**, **int**> hash\_table;
3. vector<pair<**int**, **int**> >keys;
4. **int** k;

**复杂度分析**

* **Insert**
  + **三种情况**
    - **元素element在keys（也就是动态数组）中**
      * **find函数的摊还代价是O(1)(如图a所示，查阅cppreference得到的时间复杂度，下同，并不在截图展示)**
      * **修改动态数组中元素时间复杂度是O(1)**

****

图表 a：cppreference给出的官方解释

* + - **元素不在keys中并且keys.size()<k-1**
  + 将元素插入keys中，并且设置second为1。使用vector中的push\_back操作实现，时间复杂度为O(1)
  + 在哈希表中插入元素以及所在位置的映射。使用unorodered\_map的insert函数和vector的size函数，unordered\_map的insert函数摊还代价是O(1)的，vector的size函数摊还代价也是O(1)
    - **元素不在keys中并且keys.size()=k-1**
  + 需要遍历keys中k-1个元素将其second减一，如果等于0就移除
  + 移除的时候，同时需要维护hash\_table中对应元素
  + 对second减1、unordered\_map的erase、vector的pop\_back和back都是O(1)的，所以这个操作的时间复杂度是O（k）

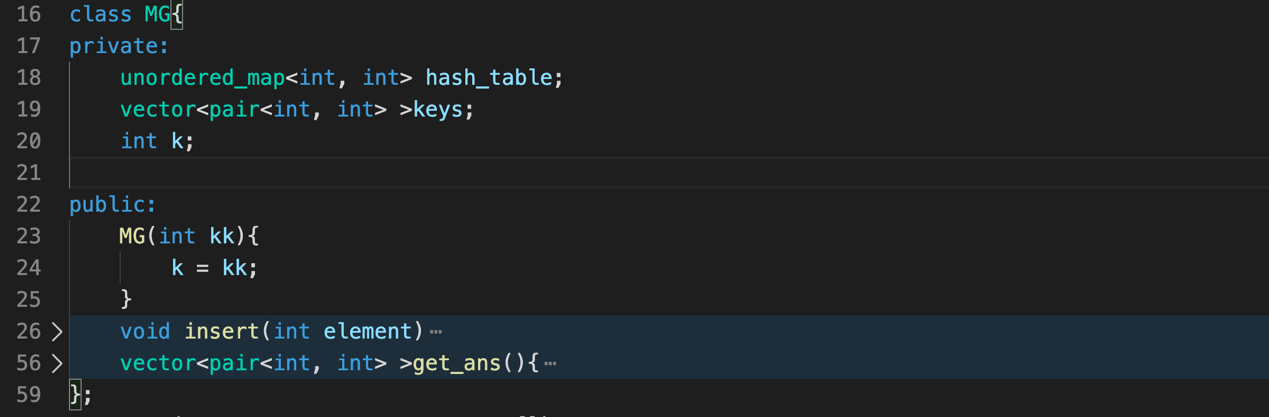
1. **void** insert(**int** element)
2. {
3. auto it = hash\_table.find(element);
4. assert(keys.size() <= k-1);
5. //元素在k-1个中
6. **if** (it != hash\_table.end())
7. {
8. keys[it->second].second++;
9. }
10. **else** **if** (keys.size() < k - 1)
11. {
12. keys.push\_back(pair<**int**, **int**>(element, 1));
13. hash\_table.insert(pair<**int**, **int**>(element, keys.size()-1));
14. }
15. **else**
16. {
17. **for**(**int** i = 0;i<keys.size();i++){
18. keys[i].second--;
19. **if**(keys[i].second==0){
20. hash\_table.erase(keys[i].first);
21. auto temp = keys.back();
22. keys.back() = keys[i];
23. keys[i] = temp;
24. hash\_table[keys[i].first] = i;
25. keys.pop\_back();
26. i = i-1;//进行这个被交换过来对元素的更新
27. }
28. }
29. }
30. }

# 三、程序设计框架：

## （1）核心类的名称及功能;

Class MG

* Hash\_table对应上文中的哈希表
* Keys对应上文中的动态数组
* K是MG算法中的k
* Insert操作插入元素
  + 按照上文对三种情况分别进行处理
* Get\_ans函数返回mg算法得到的结果



## （2）各核心类、各核心函数之间的关系；

* 数据被储存在raw\_data中
* Main函数中创建MG对象，并按照顺序将raw\_data中的每一个元素insert到对象中
* 结束以后返回结果即为MG算法的结果

## （3）输入输出的格式等

### 输入

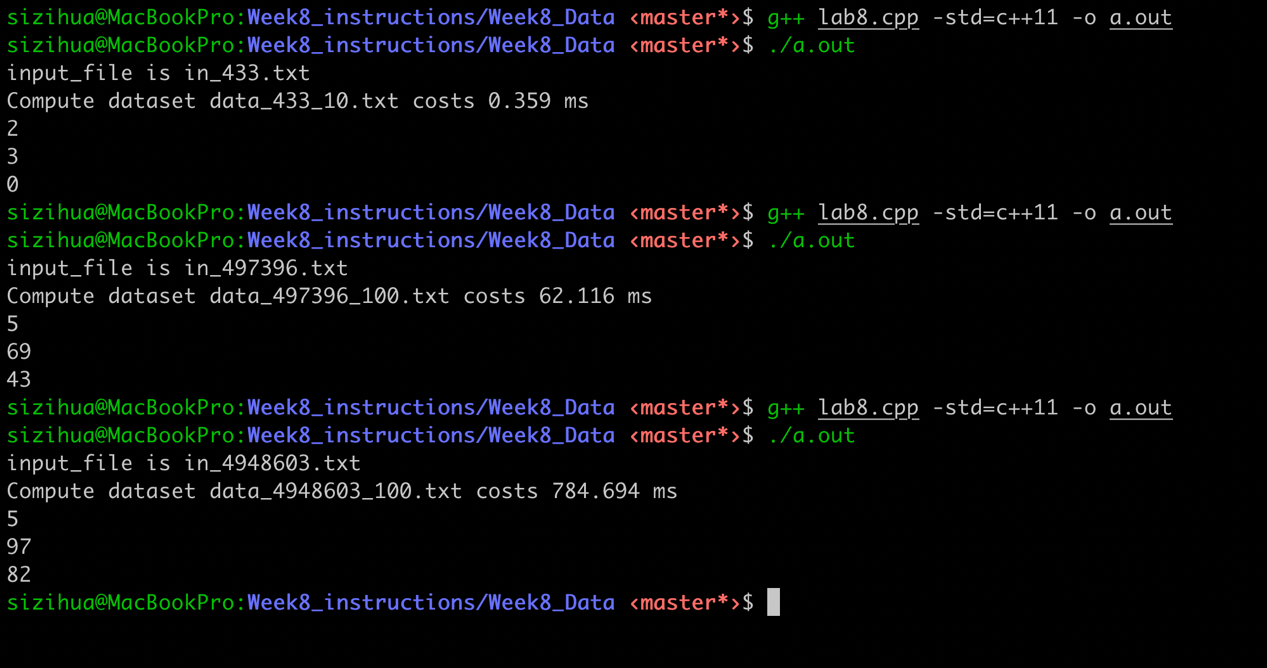
* 在62和63行的file\_name保存读入数据，input\_file保存.in文件。
  + Read\_fileh函数中使用fstream加载数据，所以输入的文件名是相对路径或者绝对路径均可。

### 输出

* 第一行为当前读入文件的信息
* 第二行为运行时间
* 接下来是.in要求的输出结果

# 四、实验结果说明：

## 实验结果截图（请保证截图清晰、美观、大小合理）；



可以看到结果在误差范围内