

**Lab Report**

**实验报告**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2023-2024** |
| **Major**: | Software Engineering |
| **Class**: | 2022 |
| **Student Name**: | 吴孜远 |
| **Student ID:** | 222022321062009 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Hashing and Spell-Checking  散列（哈希）与拼写检查 | | | |
| Date | | Dec，2023 | Type | | ☑Confirmatory （验证确认型）  ☑Design（设计型）  🗆Comprehensive（综合型） |
| 1. **Objective & Requirements（实验目的）**    1. Understand the basic principles of hashing   理解散列的基本原理   * 1. Grasp the implementation of hash map, in particular, the design of hash function and the different collision resolution strategies (chaining, open addressing, etc.)   掌握散列的实现技术，尤其是散列函数的设计，以及各种不同的冲突消解策略（链式散列、开放地址散列，等）   * 1. Graph the various operations on hash map, including insertion, deletion, finding, etc.; can perform theoretical complexity analysis for these operations.   掌握散列表上的常见操作，如插入、删除、查找；能够从理论上分析这些操作的复杂度   * 1. Learn how to use hash map in real applications, e.g. complier symbol table, spell checker, etc.   学习使用散列表解决实际问题，如编译器符号表、拼写检查等。 | | | | | |
| 1. **Experimental environment (**platform and software**)（实验环境）**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. Experimental content and design (Main Content, Procedure, Codes and Results)（此部分应包含每一个实验内容的详细设计，含实验思路、详细实验步骤、核心代码说明等）   Task 1   1. You are provided with a template container class for hash map. Some basic definitions and operations have already been defined in the class. Try to read and comprehend the codes.   阅读并理解所提供代码对hash map类的实现。   1. You are provided with a text file storing the IELTS vocabulary. Please use the provided hash map class to implement a dictionary that supports efficient lookup of words from the vocabulary. To be specific, you are provided with a hash map driver class that defines three methods, i.e.   基于所提供IELTS词汇表的文本文件，使用hash map类实现一个支持单词高效查找的词典。具体地，需要实现驱动类driver中定义的三个方法，即   * + void setUpFiles();   which opens the text file for reading the vocabulary  打开存储词汇表的文本文件以供读取（本方法已实现）  （建议将词汇表文本文件与项目的源代码文件放在同一个目录下，这样可以直接以文件名定位到该文件）   * + void readAndProcess();   which repeatedly reads each line of the vocabulary file and meanwhile set up the dictionary using hash map;  重复读取词汇表文本文件的每一行，同时构建英文词典，该词典以hash map方式存储  思路：关键问题是如何将一行中的单词和解释分成两个字符串，单词是key，解释是value，我们这里采用string类的方法substr，这个方法接收一个参数i时截取i之后的部分，输入两个参数i，j就可以截取ij之间的部分。我们遍历当前行遇到第一个空格时就分割字符串，为key和value两部分。最后将键值包装成一个key\_value\_pair，传入insert函数就好了。  void HashMapDriver::readAndProcess()  {      string curr\_line;      while (getline(inFile, curr\_line)) //repeatealy read a new line into curr\_line                                          //from input file until EOF                                          //and meanwhile set up the dictionary      {          //please implement this processing!          //每一行取空格之前的单词为key,后面的为value          int i = 0;          string key, value;          while (1) {              if (curr\_line[i] == ' ') {                  key = curr\_line.substr(0, i);                  value = curr\_line.substr(i, curr\_line.length());                  break;              }              i++;          }          //向哈希表中对应下标处存放键值对          key\_value\_pair<const string, string> p(key, value);          dict.insert(p);      } // while        inFile.close(); //close input file  } // readAndProcess   * + void testDictionary();   which reads a word from users’input, look up the word in the dictionary stored as a hash map, and then returns the Chinese meaning of the word.  从用户键盘输入读取一个单词，在词典中查找并返回中文释义  思路：这个简单，只要将输入的word当作key传入find方法就好了，find方法会返回一个迭代器itr，这个迭代器经过\*运算符返回的是一个key\_value\_pair，如何获取其value值呢，调用key\_value\_pair的second字段就好了。最后我们想要循环这个过程，那就用一个while检测word是否为QUIT。  void HashMapDriver::testDictionary()  {      system("chcp 65001"); //set the encoding of console to UTF-8 if necessary      string word;      cout << "load\_factor: " << dict.getLoad\_factor() << endl;      cout << "count: " << dict.size() << endl;      cout << "Max Comparisons: " << dict.getMaxComparisons() << endl;      cout << "Average Comparisons: " << dict.getAverageComparisons();      cout << endl << "Please input a word; input QUIT to quit." << endl;      getline(cin, word); //read a new word from user input      while (word != "QUIT") {          hash\_map<string, string, hash\_func>::iterator itr;          itr = dict.find(word);          cout << "\t" << (\*itr).second ;          cout << endl << "Please input a word; input QUIT to quit." << endl;          getline(cin, word); //read a new word from user input      }      //please implement the testing!    }  You are required to implement the readAndProcess() and testDictionary() methods; besides, for using the hash map class, you are required to implement a hash function, i.e.   * unsigned long hash\_func::operator() (const string key);   which is defined in the same file of the driver class.  你需要实现的方法是readAndProcess() and testDictionary()；此外，为正常使用hash map类，你需要设计实现散列函数类hash\_func, 该类也定义在driver.h文件中，作用是将字符串转换成无符号长整型的散列值。  简单按照如下公式转换：  “abandon”转换为  97\*1286 + 98\*1285 + 97\*1284 + 110\*1283 + 100\*1282 + 111\*1281 + 110\*1280  unsigned long hash\_func::operator() (const string key)  {      //please implement this!      long hash = 0;      for (int i = key.length()-1; i >= 0; i--) {          hash = hash + key[i] \* pow(128, i);      }      return hash;  }   1. Modify the hash\_map.h file to obtain the following information of your implemented dictionary, i.e.  * Load factor, that is, , and * The maximum and average comparsions for looking up all words in the IELTS vocabulary,   which can reflects the efficiency of your implemented dictionary.  修改hash map类以获取你所实现的英文词典的如下信息：   * 装填因子，即存储单词的数目/散列表数组的长度 * 所有IELTS词汇表中的单词在查找时的比较次数的最大值和平均值   上述两个指标可以在一定程度上反映你所实现的英文词典的查找效率。  装填因子：需要在每一次insert或者erase之后更新，一方面是count变化，另一方面可能进行了rehashing，length发生变化。  最大比较次数可以简单地看作是最大链表的长度，比较次数最大值发生在要查找这个最大链表的最后一个元素时，平均值相当于各个链表的平均长度，它的值其实和装填因子相同。  为了获取这三个值，我设置了三个getter函数：      float getAverageComparisons() {          float totalComparisons = 0;          for (int i = 0; i < length; i++) {              totalComparisons += buckets[i].size();          }          return totalComparisons / length;      }      int getMaxComparisons() {          int maxComparisons = 0;          for (int i = 0; i < length; i++) {              if (buckets[i].size() > maxComparisons) {                  maxComparisons = buckets[i].size();              }          }          return maxComparisons;      }      float getLoad\_factor() {          return load\_factor;      } | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）（此部分应包含实验结果，对实验结果的分析，实验收获的总结，实验中存在问题的讨论等；另外，需要回应一下如下思考题：1. 在基于hash map的词典上如何实现错误拼写的纠错提示，即给出一个候选单词推荐列表？2. 你在思考题1中的纠错实现复杂度是多大？能否勇其他数据结构实现效率更高的纠错提示算法？     可以看到，输入单词，可以输出其翻译。  本题雅思单词表装填因子的值为0.718，最大比较次数为38，平均比较次数与装填因子一样。  思考题：   1. 可以通过在各个位置中插入一个字母或者挨个删除各个字母生成一个很接近原单词的单词（不一定是正确拼写），然后在map中寻找，这样就找到一些可能的存在的单词。 2. 因为要对每一个位置依次尝试，所以时间复杂度应该是O(n)。至于算法，我上网搜集了一下，发现了很多比较成熟的模型或者算法：   Trie（字典），是一种树形数据结构，非常适合存储和搜索大量的单词。通过使用 Trie，你可以快速检查一个字符串是否是一个有效的单词。N-gram 模型是一种语言模型，可以分析文本中的 n 个连续的单词或字符。编辑距离算法（例如Levenshtein距离），可以使用一些优化技术，如动态规划的空间压缩，以减少计算的复杂性。这对于大规模的拼写纠错建议集合特别有帮助。 | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |