CST LAB2 HashFun

请下载附件中的数据集、实验框架和样例。

dataset.zip (attachment/ac76/ac7665ae245bb06a1717840b884ead4669694632.zip)

framework.zip (attachment/5797/5797ee0ebda08d69469cd0e4bc85f121b09ebc1e.zip)

附件下载后的名称是一串字母和数字的乱码，属于正常现象。

framework.zip 内含:

```
实验框架：
main.cpp
hashtable.cpp
hashtable.h
Makefile
用于帮助理解输入输出格式的样例数据：
1.in
1.out
```

使用Makefile不是必须的，你也可以直接用g++命令来编译多个文件组成的项目。

dataset.zip 内含:

```
poj.txt
hdu.txt
```

0. 背景知识和准备工作

“查询字符串到数字的对应关系”在实际应用中很常见。比如查通讯录，需要从姓名（字符串）对应到电话号码（数字）。我们可以使用哈希表来存储”字符串到数字的对应关系“。

哈希表有多种实现方法，性能表现不同。题目会给出一个哈希表的程序框架，你需要在框架上，实现不同的哈希表策略，构造测试数据，进行性能测试，并分析测试结果。

0.1一些帮助实验框架阅读理解的题目

（想明白这些问题可以帮助你理解实验框架，但你不需要要在实验报告中回答这些问题，它们不占分）：

1. hashtable结构体中有两个指针成员变量 `hashing_strategy* my_hashing; collision_strategy* my_collision`；如果直接用hashing_strategy和collision_strategy结构体作为hashtable的成员，会出现什么问题？（提示：强制将子类转换为基类时会发生什么？）

2. hashtable中的插入和查询都调用了冲突解决函数 `(*my_collision)( )` 分别起到什么作用?

3. `my_collision->init()` 可能有什么用处?

4. 如果插入一个已经存在的词条，现在的实验框架会如何处理?

5. 如果插入新词条时发现哈希表已满，现在的实验框架会如何处理？（**注意，你不需要修改框架在这里的处理方式，只需要在构造测试数据时，插入的次数不要太多，避免发生哈希表已满的状况**）

6. 如果进行查找时不存在对应的结果，现在的实验框架会如何处理？

7. 代码中给出了一个常量 const int TABLE_SIZE = 499883; 它是质数并且除4余3，这在哈希算法中代表什么？（**你在测试时，可以使用其他的具有类似性质的数字来作为哈希表的大小，不必局限于499883**, 通过搜索引擎可以容易地找到质数筛法的代码。）

0.2一些可能用到的函数

在生成测试数据的时候，我们可能希望将程序输出写入到某个文件，或者将某个文件作为程序的输入。C++语言中可以用 freopen()进行文件重定向。

使用freopen()需要#include<cstdio>

在main()的开头加入一句freopen("a.txt","w",stdout); 可以让程序把输出写到a.txt这个文件里（注意会覆盖文件原有的内容!)

在main()的开头加入一句freopen("b.txt","r",stdin);可以让程序把b.txt这个文件作为输入。

```
#include<cstdio>
int main(){
    freopen("a.txt","w",stdout);
    freopen("b.txt","r",stdin);
    //接下来，cin/cout/printf/scanf都会从b.txt尝试读取数据，并将输出打印到a.txt
    return 0;
}
```

使用clock()需要#include<ctime>

如果你想测量一段代码运行需要的时间, 在这段代码前面加上double t1=clock(); 这段代码后面加上double t2=clock();

运行完之后, (t2-t1)/CLOCKS_PER_SEC 就是这段代码运行的时间（单位为秒）

```
#include<ctime>
//...
double t1=clock();
/*要测时间的代码段*/myfunc();
double t2=clock();
double result = (t2-t1)/CLOCKS_PER_SEC;//result里存的就是要测时间的代码段的运行时间，单位秒
```

1. 编写数据生成器

我们提供了 `poj.txt` 和 `hdu.txt` 两个数据集，是poj.org和acm.hdu.edu.cn两个做题网站上的用户ID、排名和做题量。 `hdu.txt` 包含utf-8字符（如中文汉字等）， `poj.txt` 只包含ASCII字符。**注意：在完成这道题目时，你只需要对poj.txt做处理, hdu.txt只提供给感兴趣的同学对utf-8字符串的哈希做探究，是否用到hdu.txt不影响本题的得分**

你需要根据 `poj.txt` ，用C++编写程序作为数据生成器（数据生成器可以使用OJ上禁止使用的头文件），以 `poj.txt` 为输入时，能够输出格式类似于 `1.in` 的测试数据

需要保证对每条映射关系只进行一次插入操作。

每一行一个操作，每一行的第一个数字表示操作类型。

- 数字是0表示给出一条需要插入哈希表的“字符串-数字”映射关系，

- 数字是1表示查询某个字符串对应的数字（不存在时认为映射到-1），
- 数字是2表示输入结束。

一种方法是，编写一个程序，以 `poj.txt` 为输入，提供不同的命令行参数，可以生成不同的测试数据。

支持的命令行参数至少需要包括：测试数据中插入操作的次数、查询操作的次数。

也可以考虑将这些参数作为程序输入的一部分。也就是程序首先接受这些参数作为输入，然后再读入poj.txt，然后根据参数生成一个测试数据。

数据生成器最好具有一定的随机性，用同样的参数执行数据生成器，可以产生不同的测试数据。（例如使用<cstdlib>的rand()函数来进行随机选取，用srand()函数重置随机数种子）

生成的测试数据应当具有适当的性质，可以用来比较之后实现的各种哈希策略。为此，也可能需要在实现不同的哈希策略后，返回来修改数据生成器。

2. 不同哈希策略的实现

你需要在我们提供的哈希表框架的基础上实现不同的哈希策略。最终提交的版本中，尽量不要修改已经提供的类和函数，而是通过实现更多的strategy结构体来实现不同的哈希策略。如果要修改已经提供的类和函数，需要在实验报告中做出说明（说明哪里做了修改，并说明为什么这样修改后，不影响对不同哈希策略做公平的比较，不影响得分）。

在我们的哈希表框架中，你需要继承 `hashing_strategy` 和 `collision_strategy` 定义一些新的strategy结构体, 实现更多的哈希函数方案和冲突处理方案。

具体地：

1. 继承hashing_strategy, 实现一种针对ascii字符串(即来自poj.txt的数据）设计的哈希函数。这种哈希函数用到字符串中所有的信息，但将字符串**不均匀地**映射到哈希表中。（“坏”的哈希函数）框架中给出的哈希函数没有用到字符串中所有的字符，所以不满足要求。“用到字符串中所有的信息”可以理解为：只对字符串中一个字符做出修改，绝大多数情况下将导致哈希的结果发生改变。
2. 继承hashing_strategy, 实现一种针对ascii字符串(即来自poj.txt的数据）设计的哈希函数。这种哈希函数用到字符串中所有的信息，将字符串均匀地映射到哈希表中。 （“好”的哈希函数）
3. 继承collision_strategy, 实现双向平方试探策略。
4. 继承collision_strategy, 实现公共溢出区策略。

（关于实现公共溢出区策略的提示：可以将 `hashtable.Table[]` 的一部分作为哈希表，一部分作为公共溢出区。为此，可以修改框架中的hashtable::query()和hashtable::insert()函数，例如判断当前冲突排解策略为公共溢出区策略时，更改*my_hashing调用时传入的参数table_size。或者，为冲突排解策略增加一个成员函数，通过这个成员函数获得*my_hashing调用时传入的参数。你也可以采用其他的方法，但需要在实验报告中说明，并确保不同哈希算法比较时的公平性）

在实验报告中，简要描述你的两个哈希函数的实现思路、两个冲突处理策略的实现方法。（不得超过400字，超出太多会扣分）

对哈希函数的描述可以只用“纯数学公式”。对冲突处理的描述需要结合代码细节。

3. 进行测试

你可以对 `main.cpp` 做一些修改，从而可以使用命令行参数方便地选择采用哪种哈希函数(hashing)策略、哪种冲突处理(collision)策略。

新实现的两种哈希函数（不包括框架里最开始的哈希函数），两种新实现的冲突处理策略，加上最开始提供的冲突处理策略，有2种哈希函数、3种冲突处理策略，两两组合，有6种不同的哈希表实现。

不同数据规模、不同的插入/查询操作比例、插入和查询的不同分布方式，都可以构造出不同的测试数据。

你需要构造来自 `poj.txt` 的3组不同的数据。

然后，你需要进行 6种不同的哈希表 X 3种不同的数据，共18次测试，获取运行时间数据，汇总成文本文档，或.csv表格等便于你自己理解的格式。

如果某些程序运行完毕需要的时间过长，不必将其运行完，直接杀死程序，在结果中用“大于Y秒”表示即可。（Y是你自己设定的一个门槛）

测试时，尽量关闭电脑上的其他程序以减少干扰。哈希表的大小需要自己选择一个合适的。

你可以尝试编写脚本进行这个重复测试工作，而不是手工重复18次实验。

需要在实验报告中简单描述你的3个测试数据的构造方法、数据特征。

（不得超出300字，超出太多会扣分）

4. 分析结果

在实验报告中回答这些问题（以下题目没有标准答案，符合你的测试结果，自圆其说即可）

1. “好”和“坏”的哈希函数的性能比较情况如何？为什么是这样？
2. 线性试探和双向平方试探的性能比较情况如何？为什么是这样？
3. 在测试数据中，开放散列(公共溢出区)和封闭散列(使用试探链)的比较情况如何，谁占优势？在你的测试结果中不占优势的处理策略，处理什么样的实际数据时会更适合使用？
4. 设计哈希函数时，我们往往假定字符串每个位置上出现字符集内每个字符的概率都是相等的，但实际的数据集往往并不满足这一点。这可能造成什么影响？
5. 实验框架的哈希表容量一定，如果希望哈希表能够做到动态扩容和缩容，你准备如何实现？(介绍思路即可)

（不要超出300字，超出太多会扣分）

5. 最终提交

记得使用你的代码进行一次黑盒测试。数据范围不大，仅用于验证你的代码正确性，第一个测试数据是下发的1.in和1.out，第二个测试数据包含3000个插入操作和4000个查询操作。

（黑盒测试数据中不包含utf-8字符，只包含ASCII字符）

提交黑盒测试时，参照PA handbook的说明：代码如果含有多个文件，可以将所有代码置于顶层目录直接打包（.zip、.tar、.tar.gz 等格式）提交，目录和文件名不能有空格、中文、特殊字符。

你需要在解题报告处用一个zip压缩包提交这些内容，尽量不提交额外的内容。

1. 你所编写的数据生成器的源代码，和生成数据时使用的参数。但**不要**提交你生成的数据文件，也**不要**提交poj.txt和hdu.txt，否则会扣分。
2. 你进行修改后的实验框架，包含你实现的更多哈希函数策略、冲突排解策略，但不应改动框架中原有的函数和类。
3. 测试得到的结果。其格式应当便于理解，或对格式有说明。
4. 实验报告，包括一段”不得超出400字“的内容，和两段”不得超出300字“的内容。也可以包含其他你认为有必要的内容，如对实验框架原有函数做出改动的说明。
5. 对这道题目的感受和建议、以及估计你完成题目实际使用的时间。这一项不是必做的，也不影响得分，但未来可能会根据反馈对题目做调整。

评分时主要依据实验报告，结合代码和测试结果。按要求实现哈希策略、按要求构造测试数据并完成测试、言之成理地回答思考题，各约占三分之一分数。
