**实验四 学生学号Hush查找表（散列表）**

1. 实验题目：构造哈希查找算法，对班级学生学号进行查找，输出平均成功查找次数和平均失败查找次数。
2. 解题思路：

1）构造哈希查找表的数据结构，这里采用开列法——将冲突元素存储在线性表的后继链表中。

2）初始化哈希线性表，将每个线性表元素的指针值指控。

3）分析学号的数字特征取第4，7，8，9，10位作为特征位，由于4，7两位方差不大，对其平方，待定系数线性相加再模总数作为哈希函数。

4）根据哈希函数确定的下标，将.csv文件中的数据读入结点尾插入后继链表，完成创建。

5）查找时沿链表查找返回查找次数，求其平均。

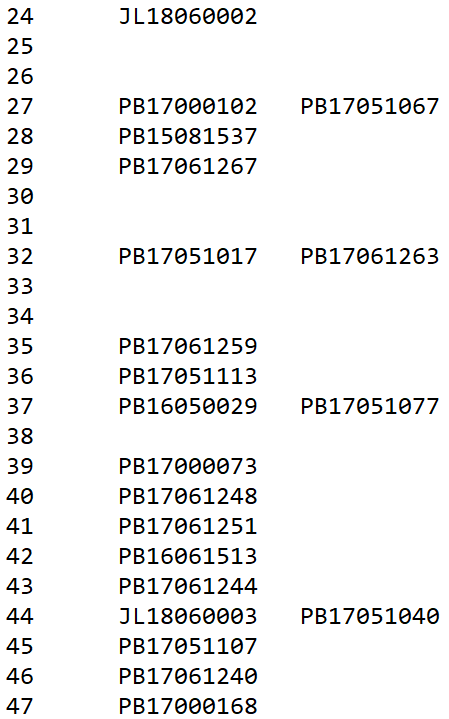
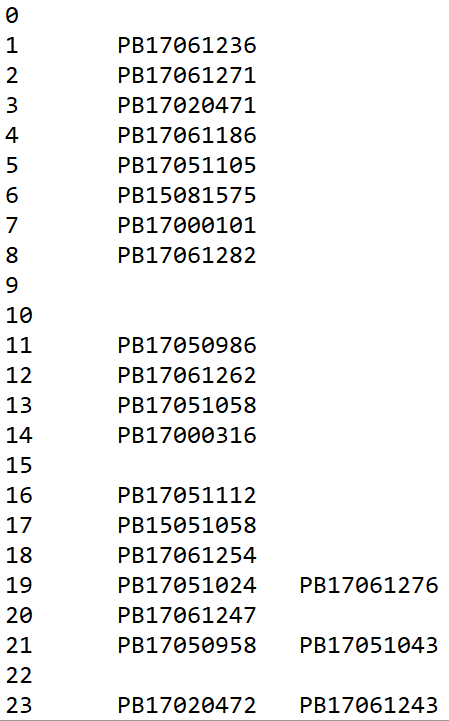
6）对哈希函数中的待定参数进行多次尝试选择合适的参数。

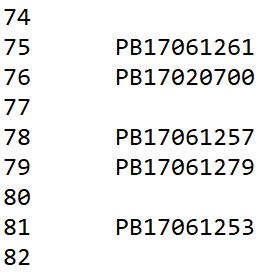
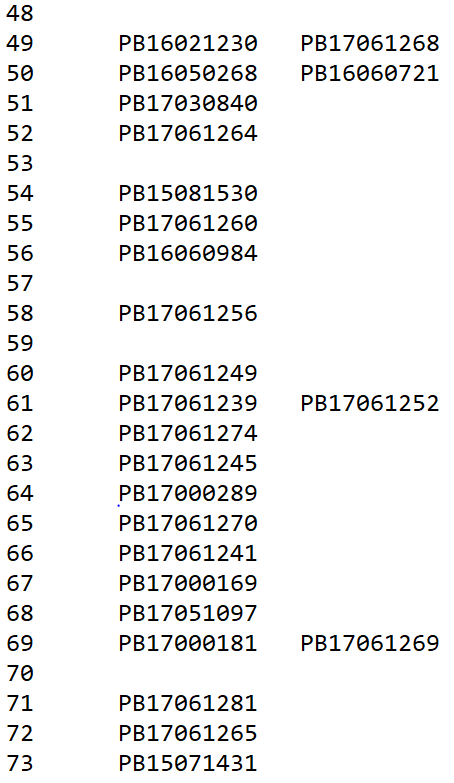
3.源代码

#include **<iostream>**#include **<string.h>**#define **n** 83  
#define **n1** 73  
**typedef struct** RAM{  
 RAM \* next;  
 **char** str[10];  
}RAM;  
  
**void** InitRAM(RAM \*& HushList) {  
 HushList = (RAM \*)malloc(**sizeof**(RAM));  
 HushList->next = **nullptr**;  
 HushList->str[0] = **'\0'**;  
}  
  
**int** Hush(**const char** str[10]) {  
 **return** ((str[6]-48)\*(str[6]-48) + (str[7]-48) + 71\*(str[8]-48)\*(str[8]-48) + 20\*(str[9]-48) + 81\*(str[3]-48))%**n**;  
}  
  
**void** print(RAM \* HushList[]) {  
 **int** i;  
 **for**(i = 0; i < **n**; i ++){  
 printf(**"%d\t"**, i);  
 RAM \* p = HushList[i];  
 **while**(p->next != **nullptr**){  
 printf(**"%s "**, p->next->str);  
 p = p->next;  
 }  
 printf(**"\n"**);  
 }  
}  
  
**void** InsertRAM(RAM \* HushList[]) {  
 FILE \* fp;  
 **if**((fp = fopen(**"test.csv"**,**"r"**)) == **NULL**){  
 printf(**"ERROR!!"**);  
 exit(0);  
 }  
 **char** str[11];  
 **int** i = 0;  
 **while**(i < **n1**){  
 RAM\* p, \*q;  
 p = (RAM \*)malloc(**sizeof**(RAM));  
 fgets(str, 11, fp);  
 **if**(fgetc(fp) == **'\r'**)  
 fseek(fp,1L,**SEEK\_CUR**);  
 q = HushList[Hush(str)]->next;  
 p->next = **nullptr**;  
 strcpy(p->str, str);  
 **if**(q == **nullptr**)  
 HushList[Hush(str)]->next = p;  
 **else**{  
 **while**(q->next != **nullptr**){  
 q = q->next;  
 }  
 q->next = p;  
 }  
 i ++;  
 }  
 fclose(fp);  
}  
  
**double** Search(RAM \* HushList[], **const char** \* file){  
 FILE \* fp;  
 **if**((fp = fopen(file,**"r"**)) == **NULL**){  
 printf(**"ERROR!!"**);  
 exit(0);  
 }  
 **char** str[11];  
 **int** i = 0;  
 **int** count = 0;  
 **char** c;  
 **while**((c = fgetc(fp)) != **EOF**){  
 fseek(fp, -1L, **SEEK\_CUR**);  
 RAM \* p;  
 fgets(str, 11, fp);  
 **if**(fgetc(fp) == **'\r'**)  
 fseek(fp,1L,**SEEK\_CUR**);  
 p = HushList[Hush(str)]->next;  
 count ++;  
 **if**(p != **nullptr**) {  
 **while**(p->next != **nullptr**) {  
 **if**(strcmp(p->next->str,str) == 1)  
 **break**;  
 count ++;  
 p = p->next;  
 }  
 }  
 i++;  
 }  
 fclose(fp);  
 **return float**(count)/**float**(i);  
}  
**int** main() {  
 **int** i;  
 **double** a, b;  
 RAM \* HushList[**n**];  
 **for**(i = 0; i < **n**; i ++){  
 InitRAM(HushList[i]);  
 }  
 InsertRAM(HushList);  
 print(HushList);  
 a = Search(HushList,**"fail.csv"**);  
 b = Search(HushList,**"success.csv"**);  
 printf(**"The failure average search time m is %lf\n"**,a);  
 printf(**"The success average search time n is %lf\n"**,b);  
 printf(**"m + n = %lf"**, a + b);  
 **for**(i = 0; i < **n**; i ++){  
 **delete** HushList[i];  
 }  
}

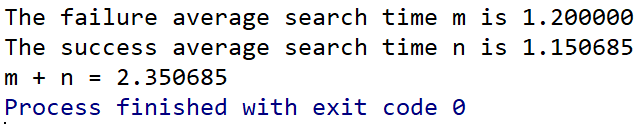
1. 运行结果

1）哈希表





2）平均查找次数



1. 心得体会
2. 用链表是接近最好的冲突解决办法，牺牲空间换时间。
3. Hush函数可以用待定系数办法，结合循环暴力推算。
4. 读取xls文件先转换成csv文件。