实验五-哈希表

PB17081504
廖洲洲
2018.12.23

1. 实验要求

a) 输入关键字序列；

b) 用除留余数法构建哈希函数，用线性探测法解决冲突，构建哈希表 HT1；

c) 用除留余数法构建哈希函数，用拉链法解决冲突，构建哈希表 HT2；

d) 分别对 HT1 和 HT2 计算在等概率情况下查找成功和查找失败的 ASL；

e) 分别在 HT1 和 HT2 中查找给定的关键字，给出比较次数

2. 实验内容

a) 建立哈希表-除留余数法

取关键字被某个不大于哈希表表长 m 的数 p 除后所得的余数为哈希地址。即

$$H(key) = key \ MOD \ p, p<=m$$

它不仅可以对**关键字直**接取模，也可在折叠、平方取中等运算之后取模。对 p 的选择很重要，一般取素数或 m，若 p 选的不好，容易产生同义词。

b) 处理冲突（这里采用了两种冲突处理方法）

- 开放寻址法：

$$Hi=(H(key) + d_i) \ MOD \ m, i=1,2,\cdots,k(k<=m-1)$$

其中 H(key）为哈希函数；m 为哈希表表长，d_i 为增量序列

使用线性探测再散列；d_i =1,2,3，…，m-1，

- 链地址法：将所有同义词的关键字存储在同一个单链表中，称这个单链表为同义词子表，在散列表中只存储同义词子表的头指针。只要有冲突，就在同义词的子表中增加结点。

c) 统计计算等概率情况下查找成功和失败的 AVL
d) 在哈希表中进行查找

3. 实验关键代码讲述

A. 线性探测法解决冲突

```
//数据的输入
printf("请输入测试文件名：\n");
scanf("%s",file);
if ((fp = fopen(file, "r")) == NULL) {
    printf("can't open this file!\n ");
    exit(0);
}
fscanf(fp,"%d",&len);
num=(int *)malloc(len*sizeof(int));
for(int i=0;i<len;i++)
    fscanf(fp,"%d",&num[i]);
printf("您输入的数据为：\n");
for(int i=0;i<len;i++)
    printf("%d  ",num[i]);
fscanf(fp,"%d",&p);
```

数据输入

```
//哈希表的建立
HT1=(int *)malloc(p*sizeof(int));
for(int i=0;i<p;i++)
    HT1[i]=-1;
for(int i=0;i<len;i++){
    Hkey=num[i]%p;
    if(HT1[Hkey]==-1)
        HT1[Hkey]=num[i];
    else{
        for(d=2,Hkey2=(Hkey+1)%p;HT1[Hkey2]!=-1&&d<=p-1;d++)
            Hkey2=(Hkey+d)%p;
        HT1[Hkey2]=num[i];
    }
}
```

建表

```
Hkey=data%p;
if(HT1[Hkey]==data){
    *times=1;
    return Hkey;
}
else{
    for(d=2,Hkey2=(Hkey+1)%p;HT1[Hkey2]!=data&&HT1[Hkey2]!=-1;d++)
        Hkey2=(Hkey+d)%p;
    *times=d;
    if(HT1[Hkey2]==-1){
        return -1;
        }
    else if(HT1[Hkey2]==data){
        return Hkey2;
    }
}
```

查找

```
printf("\n成功查找次数:  ");
for(int i=0;i<p;i++){
    if(HT1[i]==-1)
        printf("0\t");
    else{
        Search(HT1[i],&times);
        printf("%d\t",times);
        total1=total1+times;
    }
}
printf("\n失败查找次数:  ");
for(int i=0;i<p;i++){
    if(HT1[i]==-1){
        printf("1\t");\
        total2++;
    }
    else{
        for( j=i+1;j<p&&HT1[j]!=-1;j++);
        printf("%d\t",j-i+1);
        total2=total2+(j-i+1);
    }
}
```

次数统计

B. 拉链法解决冲突

```
//数据输入
printf("请输入测试文件名:\n");
scanf("%s",file);
if ((fp = fopen(file, "r")) == NULL) {
    printf("can't open this file!\n ");
    exit(0);
}
fscanf(fp,"%d",&len);
num=(int *)malloc(len*sizeof(int));
for(int i=0;i<len;i++)
    fscanf(fp,"%d",&num[i]);
```

数据输入

```
//开始建表
ChainHash=(Chain *)malloc(p*sizeof(Chain));
for(int i=0;i<p;i++)
    ChainHash[i].next=NULL;
for(int i=0;i<len;i++){
    Hkey=num[i]%p;
    chain=(Chain *)malloc(sizeof(Chain));
    chain->data=num[i];
    chain->next=ChainHash[Hkey].next;
    ChainHash[Hkey].next=chain;
}
```

建表

```
Hkey=data%p;
for(chain=ChainHash[Hkey].next;chain&&chain->data!=data;chain=chain->next)
    (*times)++;
if(chain==NULL)
    return -1;
else
    return Hkey;
```

查找

```
printf("\n成功查找次数：\n ");
for(int i=0;i<p;i++){
    if(ChainHash[i].next==NULL)
        printf("\t0");
    else
        for(chain=ChainHash[i].next;chain;chain=chain->next){
            Search(chain->data,&times);
            total1=total1+times;
            printf("\t%d",times);
            }
    printf("\n");
}
printf("\n失败查找次数：\n ");
for(int i=0;i<p;i++){
    j=1;
    for(chain=ChainHash[i].next;chain;chain=chain->next)
        j++;
    printf("%d\t",j);
    total2=total2+j;
    }
```

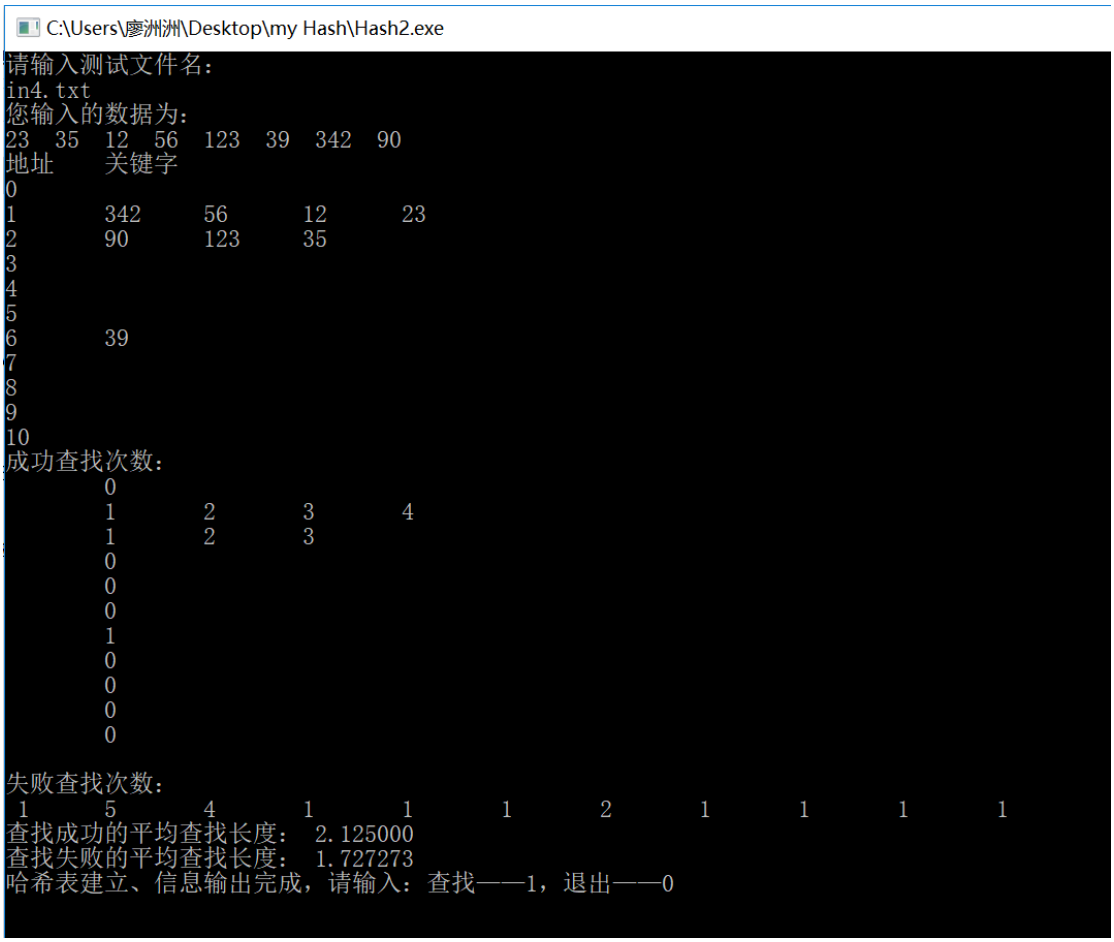
次数统计

4. 实验结果及分析

线性探测法解决冲突

```
C:\Users\廖洲洲\Desktop\my Hash\Hash.exe
请输入测试文件名:
in4.txt
您输入的数据为:
23  35  12  56  123  39  342  90
哈希表的地址：  0       1       2       3       4       5       6       7       8       9       10
表中的关键字：  -1      23      35      12      56      123     39      342     90      -1      -1
成功查找次数：  0       1       1       3       4       4       1       7       7       0       0
失败查找次数：  1       9       8       7       6       5       4       3       2       1       1
查找成功的平均查找长度：  3.500000
查找失败的平均查找长度：  4.272727
哈希表建立、信息输出完成，请输入：查找——1，退出——0
```

拉链法解决冲突



5. 实验小结

通过本次实验，我深入了解了哈希表的定义和特点，掌握了哈希函数的构造方法和解决冲突的技术，实现了哈希造表，同时学习了哈希表的查找技术，掌握了哈希表平均查找长度 ASL 的计算方法。