南開大學

《数据结构》

实验报告

实验（ 三 ）

年　　级: 2023 级

专　　业: 计算机科学与技术

姓　　名: 袁田

学　　号: 2314022

一．实验内容

实现以下内容：
1.使用链表散列实现一个散列表，关键字是整数类型，实现插入、删除、查找功能。
2.输入三个正整数序列，利用上述实现的散列表，获取三个序列的最大的元素交集。
3.输入一个整数序列，给定整数 K，获取整数序列中所有加和为 K 的不重复的连续子序列。

二．设计思想

1.先设计一个链表型的散列，用类模板实现，在第三个任务中需要实例化，是其中 data 域的数据类型变为 vector<int>。

2.在任务二中：先将第一个正整数序列的值插入到散列 h1 中，要注意数据中的整数为多位数的情况，与第二个正整数序列进行求交集，获得的数据插入到 h2 中，再将 h2 中的数据与第三个正整数序列求交集，其数据放到 h3 中。

3.在任务三中，利用前缀和数组，进行遍历将数据插入到散列 h 中，由于前缀和数组的特性，有若 k=b[i]-b[j]，则子序列为 a[j+1,…i]。

三．程序效果

程序的运行效果，输入及输出的相关要求和具体执行结果如下所示：

Part1.实现链表型散列，以整数类型的关键字实现插入删除等操作。
输入：

```
输入：# T = int
insert(2,3)
insert(3,3)
insert(3,4)
search(3)
get(3)
remove(3)
search(3)
```

输出：(由输出可看出：若 key 值相同，后输入的值会覆盖前输入的值)

```
True
4
False
```

Part2.输入三个正整数序列，获得最大的元素交集(不用去重)。
输入：
1，1，3，5
1，1，2，3
1，2，2，3
输出：
1 3
Part3.输入一个整数序列，给定整数 k，获得子序列(这段子序列的和为 k)。
输入：
1，1，1，2，1，3，4
4
输出：
1 1 2
1 2 1
1 3
4

四．核心代码

任务二：

时间复杂度：O(n)　空间复杂度：O(n)

```
//part1.实现找到三个序列中的相同子序列
Hash<int> h1;
Hash<int> h2;
string s1,s2,s3;
getline(cin,s1);
getline(cin,s2);
getline(cin,s3);

//step1.将数据输入到第一个散列列表中
for(int i=0;i<s1.size();i++)
{
    int tmp=0;
    while(s1[i]>='0'&&s1[i]<='9')
    {
        tmp=tmp*10 + s1[i]-'0';  //若输入的数据为234，则为 0*10+2=2->2*10+3=23->23*10+4=234
        i++;
    }
    if(h1.HashSearch(tmp))
    {
        int cnt=h1.Hashget(tmp);
        cnt++;
        h1.HashModify(tmp,cnt);
    }
    else
    {
        h1.HashInsert(tmp,1);
    }
}
```

时间复杂度：O(n)　空间复杂度：O(n)

```
//step2.求前两个字符串中数据的交集,将交集的数据放置到 h2中
for(int i=0;i<s2.size();i++)
{
    int tmp=0;
    while(s2[i]>='0'&&s2[i]<='9')
    {
        tmp=tmp*10 + s2[i]-'0';
        i++;
    }
    if(h1.HashSearch(tmp))
    {
        int cnt=h1.Hashget(tmp);
        if(cnt>1)
        {
            h1.HashModify(tmp,--cnt);
        }
        else
        {h1.HashDelete(tmp);}
        if(h2.HashSearch(tmp))
        {
            int count=h2.Hashget(tmp);
            count++;
            h2.HashModify(tmp,count);
        }
        else
        {h2.HashInsert(tmp,1);}
    }
    else
    {
        continue;
    }
}
```

```
//step3.将获得的交集h2与s3进行再一次求交集
for(int i=0;i<s3.size();i++)
{
    int tmp=0;
    while(s3[i]>='0'&&s3[i]<='9')
    {
        tmp=tmp*10 + s3[i]-'0';
        i++;
    }

    if(h2.HashSearch(tmp))
    {
        int cnt=h2.Hashget(tmp);
        if(cnt>1)
        {
            h2.HashModify(tmp,--cnt);
        }
        else
        {
            h2.HashDelete(tmp);
        }
        cout<<tmp<<" ";
    }
}
```

任务三：

Part1.将输入的 string 中的数据放置到 int 型的数组中

```
string s;
getline(cin,s);
int a[100]={0};
int cnt=0;
for(int i=0;i<s.size();i++)
{
    int tmp=0;
    while(s[i]>='0'&&s[i]<='9')
    {
        tmp=tmp*10 +s[i]-'0';
        i++;
    }
    if(s[i]=='-')
    {
        i++;
        while(s[i]>='0'&&s[i]<='9')
        {
            tmp=tmp*10 +s[i]-'0';
            i++;
        }
        tmp=-tmp;
    }
    a[cnt]=tmp;
    cnt++;
}
```

Part2.获得前缀和数组

```
int b[100]={0};
b[0]=a[0];
for(int i=1;i<cnt;i++)
{
    b[i]=b[i-1]+a[i];
}
```

Part3.遍历前缀和数组并在散列 h 中插入数值，分为三种情况进行讨论

1.若前缀和数组中的某个数恰好为所需 k 值，那么则输出该数对应位置前的全部数据

```
h.HashInsert(b[i],i);
int findkeys=b[i]-k;
int fir=0;
if(b[i]==k)
{
    int va=0;
    for(int j=0;j<=i;j++)
    {
        va=va*61+a[j]+10;
    }
    if(value.HashSearch(va)==false)
    {
        for(int j=0;j<=i;j++)
        {
            cout<<a[j]<<" ";
        }
        cout<<endl;
        value.HashInsert(va,0);
    }
}
```

2.由于数据中可能存在零元素，也就导致前缀和数组中 keys 为某个值的元素有重复，使用 vector 来进行容装，依次讨论。

```
if(h.HashSearch(findkeys))
{
    vector<int> p = h.Hashget(findkeys);
    int si=p.size();
    if(si>1)
    {
        for(int j=0;j<si;j++)
        {
            cout<<p[j]<<endl;
            fir=p[j]+1;
            int va=0;
            for(int sta=fir;sta<=i;sta++)
            {
                va=va*61+a[sta]+10;
            }
            if(value.HashSearch(va)==false)
            {
                for(int sta=fir;sta<=i;sta++)
                {
                    cout<<a[sta]<<" ";
                }
                value.HashInsert(va,0);
                cout<<endl;
            }
        }
    }
```

3.若该数据与零元素无关的情况

```
else
{
    fir=p[0]+1;
    int va=0;
    for(int sta=fir;sta<=i;sta++)
    {
        va=va*61+a[sta]+10;
    }
    if(value.HashSearch(va)==false)
    {
        for(int sta=fir;sta<=i;sta++)
        {
            cout<<a[sta]<<" ";
        }
        value.HashInsert(va,0);
        cout<<endl;
    }
}
```

五．总结

通过本次实验，我对于散列的各项功能有了更加深入的理解：

(1) 散列的本质其实类似于桶，数组的每个元素实际上为链表；由于 keys 和 data 的存在，散列可以起到类似于字典的作用，所有的功能(search,get…)均是通过 keys 来进行访问的

(2) 散列最重要的功能时查找，相对于普通数组和链表这种结构需要通过遍历来寻找元素，也就是时间复杂度为 O(1)；而对于散列而言，由于有 keys 的存在，可以在 O(1)的时间复杂度下查找元素

(3) Hash 函数的作用是将数据域较大的数据通过某些映射的关系可以被较小的数据域容纳，最常见的就是取模操作，大范围的数据通过取模可以缩减到低于二十位的数组进行容纳。