**散列表**

1. **实验目的和要求**

设有一个100\*100的稀疏矩阵，其中约有1%的非0元素。每个非0元素以一个三元组表示（行号，列号，元素值）。欲将此矩阵中的非0元素存放在一个如代码清单9-2所实现的闭散列表中。试设计散列表的长度、散列表中元素的类型以及元素到关键字值的转换函数。（不能使用书上的散列表类）。

**二、实验内容**

1. 实验准备
   1. 理论知识介绍
      1. 散列表
         1. 散列表提供了一种完全不同的储存和查找方法：通过将关键字值直接映射到表中的某个位置，将该关键字对应的数据元素储存在这个问之中。查找时，可以直接根据被查找的关键字值找到储存该数据元素的地址，从而获得这个数据元素。这意味着查找时间可以下降到O(1)。
         2. 散列函数

由于任何类型的数据都能转换成一个整型数，所以我们可以假设关键字的类型是整型的。

* + - * 1. 直接定址法

直接取关键字的值或关键字的某个线性函数的值作为散列地址

* + - * 1. 除留余数法

如果M时散列表的大小，关键字为x的数据元素的散列地址为：

H(x) = x mod M

* + - * 1. 数字分析法

如果在关键字集合中，每个关键字均由N位数字组成（x1,x2,…,xn）,分析关键字中的每一位数字的分布规律，并从中提取出分布均匀的若干位或他们的组合作为地址。

* + - * 1. 平方取中法

如果关键字中的各位的分布都比较均匀，但关键字的值域比数组规模大，则可以将关键字平方后，取其结果的中间各位作为散列函数值。

* + - * 1. 折叠法

如果关键字相当长，以至于和散列表的单元总数相比大得多时，可采用此方法。如果数字的分布大体上是均匀的，通常的 做法时选取一个长度后，将关键字按磁场都分组相加，抛弃进位，得到散列的结果。

* + - 1. 碰撞的解决
         1. 线性探测法

最简单可行的策略是线性探测法，即在该数组中从映射到的位置开始顺序搜索， 直到发现一个空位置。

* + - * 1. 二次探测法

在线性探测法中，碰撞会引起连锁反应，使表中形成一些较长的连续被占单元，从而使性能下降。

在二次探测法中，当发生碰撞时，它不是直接检查下一单元，而是检查远离初始探测点的某一单元，以消除线性探测中的初始聚集的问题。

它的名字是由解决碰撞时所用的公式F(i)=i2而得到的。

* + - * 1. 再散列法

再散列法中有两个散列函数H1和H2。H1用来计算探测序列的起始地址，H2用来计算下一个探测位置的步长。

1. 实验项目
   1. 分析
      1. 由于所给的矩阵的大小是100\*100的，而非0元素大概占1%，所以我们准备的散列表的长度大小大约为100，但是我们要预留位置而且散列表的长度最好是素数，所以我们所采用的散列表的长度可以为101.
      2. 散列表的元素的类型应该就是所给的矩阵中的三元组（行号，列号，元素值）。
      3. 散列表的散列函数有很多种设计方法，最简单的我们可以用除留余数法来解决。通过散列函数我们将我们初始的行号和列号转换成我们所需要的关键字值，这种转换的方法也有很多种，最简单的方法就是将行号和列号相加，那么我们所采用的散列函数的除留余数法的模就可以采用100。
   2. 方案
      1. 实现闭散列表的各个函数
      2. 构造矩阵
      3. 将矩阵中的元素放入散列表中
   3. 测试数据

int main()

{

struct MyStruct

{

int row;

int column;

int data;

};

MyStruct a[10000];

return 0;

}

**三、实验小结**

1. 重点

闭散列表的实现。

1. 难点

如何在闭散列表发生碰撞的时候进行碰撞的处理。

1. 收获、体会

散列表的碰撞的解决办法有很多，通常来说比较简单的方法一般来说效率比较低，复杂的方法效率较高。