**作业 HW5 实验报告**

姓名：陆明奇 学号：2050283 日期：2021年12月14日

1. **涉及数据结构和相关背景**

查找表是由同一类型的数据元素（或记录）构成的集合。由于“集合”中的数据元素之间存在着完全松散的关系，因此查找表是一种非常灵便的数据结构。排序是计算机程序设计中的一种重要操作，它的功能是将一个数据元素（或记录）的任意序列，重新排列成一个按关键字有序的序列。

**2. 实验内容**

**2.1 折半查找**

**2.1.1 问题描述**

二分法将所有元素所在区间分成两个子区间，根据计算要求决定下一步计算是在左区间还是右区间进行；重复该过程，直到找到解为止。二分法的计算效率是O(logn),在很多算法中都采用了二分法，例如：折半查找，快速排序，归并排序等。折半查找要求查找表是有序排列的，本题给定已排序的一组整数，包含重复元素，请改写折半查找算法，找出关键字key在有序表中出现的第一个位置或最后一个位置，保证时间代价是O(logn)。若查找不到，返回-1。

**2.1.2 基本要求**

第1行输入一个正整数n,表示查找表的长度;

第2行输入n个有序排列的整数，以空格分割

后面若干行输入为查找的方式ope待查找的整数x

若ope为lower，则表示查找元素在数列中首次出现的位置

若ope为upper，则表示查找元素在数列中最后一次出现的位置

若ope为done，则表示查找结束

输出查找元素在有序表中的位置，有序表从0开始存储。若查找不到，返回-1。

**2.1.3 数据结构设计**

采用了一个int型数组来存储要查找元素所在的集合。因为数组元素的下标代表了元素所在数组中的位置，比较方便使用二分法。

**2.1.4功能说明（函数、类）**

int Search\_Bin(char\* s, int key)

功能：查找关键字为key的元素在数组中首次或最后一次出现的位置

Para s：判断查找元素首次出现的位置还是最后一次出现的位置

Para key：待查找元素的关键字

**2.1.5 调试分析（遇到的问题和解决方法）**

二分查找需要在最初设置两个指针left和right分别指向数组元素的首元素和最后一个元素，将mid指针设置为指向数组中正中间的元素。比较mid指针指向的元素和待查找元素的大小，若小于待查找元素，则需要将left指针指向当前mid指针指向的元素。这是实现二分查找的关键，即每进行一次查找，就更新一次left或right指针的值。

**2.1.6 总结和体会**

折半查找的查找过程是：先确定待查记录所在的范围（区间），然后逐步缩小范围直到找到或找不到该记录为止。算法复杂度为O（logn）

**2.2 二叉排序树**

**2.2.1 问题描述**

二叉排序树BST（二叉查找树）是一种动态查找表，或者是一棵空树，或者是具有下列性质的二叉树：

（1）每个结点都有一个作为查找依据的关键字(key)，所有关键字的键值互不相等。

（2）左子树(若非空)上所有结点的键值都小于它的根结点的键值。

（3）右子树(若非空)上所有结点的键值都大于它的根结点的键值。

（4）左子树和右子树也是二叉排序树。

二叉排序树的基本操作集包括：创建、查找，插入，删除，查找最大值，查找最小值等。本题实现一个维护整数集合（允许有重复关键字）的BST，并具有以下功能：1. 插入一个整数 2.删除一个整数 3.查询某个整数有多少个 4.查询最小值 5. 查询某个数字的前驱

**2.2.2 基本要求**

输入：

第1行一个整数n，表示操作的个数；

接下来n行，每行一个操作，第一个数字op表示操作种类：

若op=1，后面跟着一个整数x，表示插入数字x

若op=2，后面跟着一个整数x，表示删除数字x（若存在则删除，否则输出None，若有多个则只删除一个），

若op=3，后面跟着一个整数x，输出数字x在集合中有多少个（若x不在集合中则输出0）

若op=4，输出集合中的最小值（保证集合非空）

若op=5，后面跟着一个整数x，输出x的前驱（若不存在前驱则输出None，x不一定在集合中）

输出：

一个操作输出1行（除了插入操作没有输出）

**2.2.3 数据结构设计**

由于题目要求是建立二叉排序树，因此本题数据结构采用链式二叉树。

**2.2.4功能说明（函数、类）**

bool InsertBST(BiTree& T,int key)

功能:在二叉排序树T中插入一个关键字为key的元素

Para T：指向二叉排序树根结点的指针

Para key：要插入元素的关键字

返回值：插入是否成功

bool Delete(BiTree& T)

功能:在二叉排序树中删除T指向的元素

Para T：指向待删除结点的指针

返回值：删除是否成功

bool DeleteBST(BiTree& T, int key)

功能:在二叉排序树T中删除一个关键字为key的元素

Para T：指向二叉排序树根结点的指针

Para key：待删除元素的关键字

返回值：删除是否成功

int Getnum(BiTree T, int key)

功能:在二叉排序树T中查询关键字为key的元素的数量

Para T：指向二叉排序树根结点的指针

Para key：待查询元素的关键字

返回值：查询关键字为key的元素的数量

int Getmin(BiTree T)

功能:在二叉排序树T中查询关键字最小的元素

Para T：指向二叉排序树根结点的指针

返回值：关键字最小的元素的值

BiTree Getpri(BiTree T, int key)

功能:在二叉排序树T中查询关键字为key的元素的前驱

Para T：指向二叉排序树根结点的指针

Para key：元素的关键字

返回值：指向前驱的指针

**2.2.5 调试分析（遇到的问题和解决方法）**

题目中提到会有重复的关键字，我一开始的想法是建立两个结点来分别存储关键字相同的这两个元素，但这会在排序中产生错误。之后，我选择在结点中多加入一个int型元素num，用来表示有重复关键字的元素的数量。这样一来所有关键字相同的元素就存储在了同一个结点中，排序问题得到了解决，保护了二叉排序树的性质。

**2.2.6 总结和体会**

二叉排序树的难点在于删除一个结点后仍然保持二叉排序树的性质。当被删除结点的左右子树均不空时，需要找到被删除结点左子树中的最大值，用该最大值填充被删除的结点，并重新连接好该最大值的左右子树。

**2.3 哈希表**

**2.3.1 问题描述**

哈希表（hash table，散列表）是一种用于以常数平均时间执行插入、删除和查找的查找表，其基本思想是：找到一个从关键字到查找表的地址的映射h（称为散列函数），将关键字key的元素存到h(key)所指示的存储单元中。当两个不相等的关键字被散列到同一个值时称为冲突，产生冲突的两个（或多个）关键字称为同义词，冲突处理的方法主要有：开放定址法，再哈希法，链地址法。本题针对字符串设计哈希函数。假定有一个班级的人名名单，用汉语拼音（英文字母）表示。要求：首先把人名转换成整数，采用函数h(key)=((...(key[0] \* 37+key[1]) \* 37+...)\*37+key[n-2] )\* 37+key[n-1]，其中key[i]表示人名从左往右的第i个字母的ascii码值(i从0计数,字符串长度为n，1<=n<=100)。采取除留余数法将整数映射到长度为P的散列表中，h(key)=h(key)%M，若P不是素数，则M是大于P的最小素数，并将表长P设置成M。采用平方探测法（二次探测再散列）解决冲突。（有可能找不到插入位置，当探测次数>表长时停止探测）

**2.3.2 基本要求**

输入：

第1行输入2个整数N、P，分别为待插入关键字总数、散列表的长度。若P不是素数，则取大于P的最小素数作为表长。

第2行给出N个字符串，每一个字符串表示一个人名

输出：

在1行内输出每个字符串插入到散列表中的位置，以空格分割，若探测后始终找不到插入位置，输出一个'-'。

**2.3.3 数据结构设计**

采用一个int型数组来表示哈希表的每个位置有没有被占据

**2.3.4功能说明（函数、类）**

bool IsPrime(int n)

功能：判断n是不是素数

Para n：待判断的数

返回值：n是否为素数

int CreateHash(char\* s)

功能：将字符串s存储到哈希表中

Para s：输入的字符串

返回值：字符串s在哈希表中存储的位置

**2.3.5 调试分析（遇到的问题和解决方法）**

在判断一个数n是否为素数时，我的做法是遍历从2到n-1的所有数，如果其中有数能整除n，那么n就不是素数。这样的算法需要对1作单独的判断，1本身也是一个素数。最初写代码的时候没有考虑到1。其次，需要将哈希表的长度开得稍微大一点，因为若输入的p不是素数，则哈希表的长度为大于p的最小素数m，m和p在数据很大时会相差很大。一开始数组长度开得不够大导致了越界错误。

**2.3.6 总结和体会**

哈希表的构造方法有直接定址法，数学分析法，平方取中法，折叠法，除留余数法和随机数法等。处理冲突的方法有开放定址法，再哈希法，链地址法，建立一个公告溢出区等。

**2.5 求逆序对数**

**2.5.1 问题描述**

对于一个长度为N的整数序列A，满足i < j 且 Ai > Aj的数对(i,j)称为整数序列A的一个逆序。请求出整数序列A的所有逆序对个数

**2.5.2 基本要求**

输入包含多组测试数据，每组测试数据有两行

第一行为整数N(1 <= N <= 20000)，当输入0时结束

第二行为N个整数，表示长为N的整数序列

每组数据对应一行，输出逆序对的个数

**2.5.3 数据结构设计**

采用一个int型数组来存储输入的整数序列，采用另一个数组来存储已经排好序的整数序列。

**2.5.4功能说明（函数、类）**

void msort(int b, int e)

功能：对数组中位置b-e的数进行排序

Para b：进行排序的数的起始位置

Para e：进行排序的数的终止位置

返回值：无返回值

**2.5.5 调试分析（遇到的问题和解决方法）**

归并排序分为三个步骤：将数列划分为两部分；递归地分别对两个子序列进行归并排序；合并两个子序列。其前两步都很好实现，难点在于如何合并两个子序列。注意到两个子序列在第二步中已经保证了都是有序的，第三步实际上只要把两个有序的序列合并起来就行了。

**2.5.6 总结和体会**

归并排序是一种采用了分治思想的排序算法。它是一种稳定的排序算法。它的时间复杂度为O（nlogn），空间复杂度为O（n）。归并排序的优点是排序速度快且稳定，缺点是需要很大的辅助空间，空间复杂度高。

**3.实验总结**

对查找表进行的操作有：（1）查询某个特定的数据元素是否在查找表中；（2）检索某个特定数据元素的各种属性；（3）在查找表中插入一个数据元素；（4）在查找表中删去某个数据元素。若对查找表只作前两种统称为“查找“的操作，则称此类查找表为静态查找表。若在查找过程中同时插入查找表中不存在的数据元素，或者从查找表中删去已存在的某个数据元素，则称此类查找表为动态查找表。内部排序的方法很多，但就其全面性能而言，很难提出一种被认为是最好的方法，每一种方法都有各自的优缺点，适合在不同的环境（如记录的初始排列状态等）下使用。如果按排序过程中依据的不同原则对内部排序进行分类，则大致可分为插入排序，交换排序，选择排序，归并排序和计数排序等五类；如果按内部排序过程中所需的工作量来区分，则可分为三类：（1）简单的排序算法，其时间复杂度为O（n2），（2）先进的排序算法，其时间复杂度为O（nlogn），（3）基数排序，其时间复杂度为O（d\*n）