HW5 搜索和排序 实验报告

姓名：徐恒 学号：2050035 日期：2022 年 6 月 29 日

1. 涉及数据结构和相关背景

排序：将一组杂乱无章的数据排列成一个按关键字有序的序列。
数据表(datalist):它是待排序数据对象的有限集合。
关键字(key):通常数据对象有多个属性域，即多个数据成员组成，其中有一个属性域可用来区分对象，作为排序依据。该域即为关键字。每个数据表用哪个属性域作为关键字，要视具体的应用需要而定。即使是同一个表，在解决不同问题的场合也可能取不同的域做关键字。
主关键字: 如果在数据表中各个对象的关键字互不相同，这种关键字即主关键字。按照主关键字进行排序，排序的结果是唯一的。
次关键字: 数据表中有些对象的关键字可能相同，这种关键字称为次关键字。按照次关键字进行排序，排序的结果可能不唯一。
排序算法的稳定性: 如果在对象序列中有两个对象 r[i]和 r[j]，它们的关键字 k[i] == k[j]，且在排序之前，对象 r[i]排在 r[j]前面。如果在排序之后，对象 r[i]仍在对象 r[j]的前面，则称这个排序方法是稳定的，否则称这个排序方法是不稳定的。

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

输入：
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

折半查找的基本思想：减少查找序列的长度，分而治之地进行关键字的查找。他的查找过程是：先确定待查找记录的所在的范围，然后逐渐缩小查找的范围，直至找到该记录为止（也可能查找失败）。

2.1.4 功能说明（函数、类）

void Clear() //清空数组
void PushBack(int elem) //在数组尾部插入一个元素
int UpperBound(int elem) //寻找元素最后一个出现的下标
int LowerBound(int elem) //寻找元素第一个出现的下标
int RandElem() //在数列中随机取元素
void PrintList() //打印整个数列

2.1.5 调试分析（遇到的问题和解决方法）

2.1.6 总结和体会

2.2 二叉树排序

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

(1)插入
a.插入过程比较简单，首先判断当前要插入的值是否已经存在二叉排序树中，如果已经存在，则直接返回；如果不存在，则转 b;
b.当前要插入的值不存在，则应找到适当的位置，将其插入。注意插入的新节点一定是叶子节点;
(2)删除
a.和插入一样，要删除一个给定值的节点，首先要判断这样节点是否存在，如果已经不存在，则直接返回；如果已经存在，则获取给定值节点的位置，根据不同情况进行删除、调整，转 b;
b.如果待删节点只有左子树（只有右子树），则直接将待删节点的左子树（右子树）放在待删节点的位置，并释放待删节点的内存,否则转 c
c.如果待删节点既有左子树又有右子树，此时的删除可能有点复杂，但是也比较好理解。就是在待删节点的左子树中找到值最大的那个节点，将其放到待删节点的位置。
(3)查询
查询过程比较简单，首先将关键字和根节点的关键字比较，如果相等则返回节点的位置(指针)；否则，如果小于根节点的关键字，则去左子树中继续查找；如果大于根节点的关键字，则去右子树中查找；如果找到叶子节点也没找到，则返回 NULL。查询过程的最好情况就是上面的图中那样，节点在左右子树中分布比较均匀，此时查找的时间复杂度为 O(logn)；最坏的情况就是在建立二叉排序树时，输入的关键字序列正好是有序的，此时形成的二叉排序树是一棵单支二叉树，此时查找退化成了单链表的查找，时间的复杂度为 O(n).
(4)遍历
由上面二排序树的定义可知，左子树的所有值均小于根节点，右子树的所有值均大于根节点，而这个特点正好和二叉树的中序遍历中--左子树->根节点->右子树不谋而合，所以对二叉排序树进行中序遍历得到的正好是一个有序的

2.2.4 功能说明（函数、类）

```
void UpdateHeight()
bool Find(int elem, NODE*& last, NODE*& thiss, NODE*& next, bool& found)
void Insert(int elem)  //插入一个整数
bool Delete(int elem)  //删除一个整数
int Count(int elem)  //查询某个整数有多少个
int FindMinElem()  //查询最小值
int FindFrontElem(int elem)  //查询某个数字的前驱
```

2.2.5 调试分析（遇到的问题和解决方法）

2.2.6 总结和体会

对二叉排序树有了更深刻的认识和理解

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)
注意：计算 h(key)时会发生溢出，需要先取模再计算。

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

通过哈希表减少地址的冲突

2.3.4 功能说明（函数、类）

int Insert(const std::string& str)
static int CalculateNextPrimeNumber(const int P)
static std::string RandomName(int length)
static int Square(int n)
int Hash(const std::string& str) const

2.3.5 调试分析（遇到的问题和解决方法）

2.3.6 总结和体会

数字分析法，平方取中法，折叠法，除留余数法

2.4 求逆序对数

2.4.1 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

2.4.2 基本要求

输入
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
输出
每组数据对应一行，输出逆序对的个数

2.4.3 数据结构设计

类似于归并排序算法。先将数组从中间分成两个部分，然后分别递归左半部分和右半部分，再合并排好序的左右两个部分，从而统计逆序对数。

2.4.4 功能说明（函数、类）

void mergesort(int l, int r)
void mergearray(int l, int r)

2.4.5 调试分析（遇到的问题和解决方法）

一开始直接遍历序列判断逆序数，然而超时，后续用分治方法求出逆序数

2.4.6 总结和体会

题目较为简单。