计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：朱龙康 学号：1824120002 专业：计算机科学与技术\_ 年级：\_2018级

课程：数据结构 主讲教师：\_袁彩虹\_ 辅导教师：\_\_袁彩虹\_\_

实验时间：\_\_2020\_年 \_6\_月 \_\_10\_日 \_上\_午\_8\_时至\_10\_时，实验地点\_\_\_\_\_\_\_

实验题目： 查找和排序的实现

实验目的： （1）掌握折半查找和二叉排序树两种查找方法；

（2）掌握各种不同的排序算法。

实验环境（硬件和软件） CodeBlocks

实验内容：

*（（1）编程实现两种查找方法：折半查找和二叉排序树。若查找成功，返回元素在有序数组中的位置和查找次数；若查找失败，返回出错标志和查找次数；*

*（2）在9种排序算法（直接插入排序、折半插入排序、希尔排序、冒泡排序、快速排序、简单选择排序、堆排序、归并排序和基数排序）中选择5种排序算法进行编程实现。*实验步骤：

**//------折半查找**

**int Search\_Bin(List l,int key,int &sum){//函数返回值用于返回元素位置，-1为未找到**

**//sum用于带出查找次数 初始值为0**

**int low=1,high=l.length;**

**int mid;**

**while (low<=high){**

**mid=(low+high)/2;**

**if (key==l.elem[mid]){sum++;return mid;}//找到要查找的值**

**else if (key<l.elem[mid]){sum++;high=mid-1;}//在左子表查找**

**else{sum++;low=mid+1;}//在右子表查找**

**}**

**return -1;**

**}**

**//------------二叉排序树**

**//二叉排序树的二叉链表存储表示**

**typedef struct BSTNode**

**{**

**int data;**

**struct BSTNode \*lchild,\*rchild;**

**}BsTNode,\*BSTree;**

**//二叉排序树的插入**

**void InsertBST(BSTree &T,int e){**

**if(!T) //如果树为空**

**{**

**BSTree s=new BSTNode;**

**s->data=e;**

**s->lchild=s->rchild=NULL;**

**T=s;**

**}**

**else if(e<T->data)InsertBST(T->lchild,e);//若插入值比当前节点小 则插入左子树**

**else if(e>T->data)InsertBST(T->rchild,e);//若插入值比当前节点大 则插入右子树**

**}**

**//二叉排序树的创建**

**void CreatBST(BSTree &T){**

**T=NULL; //二叉树初始化为空树**

**int e;**

**cin>>e;**

**while (e!=-1){//输入-1 代表结束**

**InsertBST(T,e); //将该元素插入二叉排序树T中**

**cin>>e;**

**}**

**}**

**//二叉排序树的查找**

**BSTree SearchBST(BSTree T,int key,int &sum){//sum为比较次数**

**if(!T||T->data==key){sum++;return T;}//查找结束**

**else if(key<T->data){sum++;return SearchBST(T->lchild,key,sum);}//在左孩子中查找**

**else {sum++;return SearchBST(T->rchild,key,sum);}//在右孩子中查找**

**}**

**//---------插入排序**

**void InsertSort(List &L){**

**for(int i=2;i<L.length;i++){**

**if(L.elem[i]<L.elem[i-1]){**

**L.elem[0]=L.elem[i]; //设置监视哨**

**L.elem[i]=L.elem[i-1]; //后移**

**}**

**int j;**

**for (j = i-2; L.elem[0]<L.elem[j]; --j) {//从后向前寻找插入位置**

**L.elem[j+1]=L.elem[j]; //后移，知道找到插入位置**

**}**

**L.elem[j+1]=L.elem[0]; //将原来的elem[i]插入**

**}**

**}**

**//----------折半插入排序**

**void BInsertSort(List &L){**

**for(int i=2;i<L.length;i++){**

**L.elem[0]=L.elem[i];//设置监视哨**

**int low=1,high=i-1;**

**while (low<=high){ //利用二分查找 找到第一个比elem[i]小的元素位置**

**int m=(low+high)/2;**

**if(L.elem[0]<L.elem[m])high=m-1;**

**else low=m+1;**

**}**

**for (int j = i-1; j >=high+1; --j) {//后移元素**

**L.elem[j+1]=L.elem[j];**

**}**

**L.elem[high+1]=L.elem[0];//将原来的elem[i]插入**

**}**

**}**

**//-----------希尔排序**

**void ShellInsert(List &L,int dk){**

**for (int i = dk+1; i <L.length ; ++i) {**

**if(L.elem[i]<L.elem[i-dk]){ //将elem[i[插入有序增量子表**

**L.elem[0]=L.elem[i]; //暂存在elem[0]**

**for (int j = i-dk; j >0&&L.elem[0]<L.elem[j] ; j-=dk) {**

**L.elem[j+dk]=L.elem[j]; //记录后移 知道找出插入位置**

**L.elem[j+dk]=L.elem[0]; //将elem[0]即原来的r[i]插入到正确位置**

**}**

**}**

**}**

**}**

**void ShellSort(List &L,int dt[],int t){**

**for (int i = 0; i <t ; ++i) {**

**ShellInsert(L,dt[i]); //一趟增量为dt[t]的希尔插入排序**

**}**

**}**

**//--------冒泡排序**

**void BubbleSort(List &L){**

**int m=L.length-1;int flag=1; //flag用于标记某一趟排序是否发生交换**

**while ((m>0)&&flag==1){**

**flag=0; //flag置为0，如果本趟排序没有发生交换，则不会执行下一趟排序**

**for (int i = 1; i <=m; ++i) {**

**if(L.elem[i]>L.elem[i+1])**

**{**

**flag=1; //flag置1 表示本趟排序发生交换**

**int t=L.elem[i];L.elem[i]=L.elem[i+1];L.elem[i+1]=t;**

**}**

**--m;**

**}**

**}**

**}**

**//-------选择排序**

**void SelectSort(List &L){**

**for (int i = 1; i < L.length; ++i) {//在L中选择关键字最小的记录**

**int k=i;**

**for (int j = i+1; j < L.length; ++j) {**

**if(L.elem[j]<L.elem[k])k=j;//k指向此趟排序中关键字最小的记录**

**if(k!=i)**

**{int t=L.elem[i];L.elem[i]=L.elem[k];L.elem[k]=t;}//交换elem[i]与elem[k]**

**}**

**}**

**}**

问题讨论：

哪种排序算法最好？