**江西师范大学计算机信息工程学院学生实验报告（9）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **专业：** | **数据科学与大数据技术2班** | **姓名：** | **赖丽婷** | **学号：** |  | **日期：** | **2021.12.09** |

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 数据结构 | 实验室名称 | 计算机综合实验室 |
| 实验名称 | 检索 | | |
| 指导教师 |  | 成绩 |  |

**1.实验目的**（结出本次实验所涉及并要求掌握的知识点）

1. 掌握顺序表的查找方法，尤其是二分查找方法，并能给予实现。
2. 掌握基于链表的查找方法，并能给予实现。
3. 掌握二叉排序树的建立及查找算法，并能给予实现。

**2.实验内容**（结出实验内容具体描述）

1. 数组顺序查找算法
2. 单链表的顺序查找算法
3. 数组的非递归查找算法
4. 数组的递归查找算法
5. 建立二叉排序树

**3.算法描述及实验步骤**（用适当的形式表达算法设计思想与算法实现步骤）

1. 数组的顺序查找:循环数组判断当前索引的数组值是否等于key->如果不等于则继续往后查找->如果等于则跳出循环->判断数组查找是否越界，越界返回-1，没有则返回跳出循环的适应
2. 单链表的顺序查找:获取不带头节点单链表的第一个结点->循环单链表，如果当前指针指向的位置不为空，判断当前指针的data是否为key->如果不为key，往后查找，找到找到或者找完->直接返回跳出循环的指针
3. 非递归的二分查找:获取数组长度，从最左位0与最右边n-1开始二分查找->找到中间位mid = (left+right)/2->判断中间位的数与key的关系，如果等于返回mid,如果大于key,则从原来的左边位到右边位为mid-1继续查找，如果小于，则从左边位为mid+1到原来右边位继续查找，直到找到或者left>right->如果最后left>right返回-1;
4. 递归的二分查找:函数中传入left与right,key->如果left>right返回-1->判断key与a[mid]值的关系，如果等于返回mid，如果a[mid]大于key调用自己函数right改为mid-1，如果小于left改为right+1;
5. 建立一颗二叉排序树:遍历输入的数据->将当前的数据与当前二叉树以有的数据进行查找->如果输入数据等于二叉树已有的数据退出循环，不匹配则去查找要插入的位置->找到位置之后给输入的数据分配空间->如果二叉树为空则输入数据放在根节点，如果不为空则判断输入数据与插入位置数据的关系->如果小于插入到左子树，如果大于则插入到右子树->循环操作直到所有数据遍历完成

**4.调试过程及运行结果**（详细记录在调试过程中出现的问题及解决方法。记录实验执行的结果）

**5. 总结**（对实验结果进行分析，问题回答，实验心得体会及改进意见）

1. 顺序查找查找成功的平均查找长度为(n+1)/2查找失败长度为(n)
2. 二分查找成功平均查找长度为log2(n+1);
3. 在平衡二叉树最好的情况下查找成功的平均查找长度为log2(n)
4. **附录**（程序源代码等）

#include "ArrayIo.h"

#define N 10000 /\*数据量\*/

/\*请将本函数补充完整，并进行测试\*/

int seqsearch(int a[], int n, int key)

{

int i = 0;

while (i < n && a[i] != key)

{

i++;

}

if (i == n)

{

return -1;

} else

{

return i;

}

}

int main()

{

setbuf(stdout, NULL);

int a[N], n, x, pos;

clock\_t start, finish;

n = readData(a, N, "../data1.txt"); /\*从data1.txt文件中读入前N个数存入a中，函数返回成功读入的数据个数\*/

printf("请输入要查找的整数：");

scanf("%d", &x);

start = clock();

pos = seqsearch(a, n, x);

finish = clock();

if (pos == -1)

printf("查找失败\n");

else

printf("a[%d]=%d\n", pos, x);

printf("%ld\n", finish - start);

return 0;

}

#include "slnklist.h"

#include "time.h"

#define N 500000 /\*数据量\*/

/\*请将本函数补充完整，并进行测试\*/

linklist seqsearch(linklist head, int key)

{

linklist p;

p = head->next;

while(p && p->info != key)

{

p = p->next;

}

return p;

}

int main()

{

setbuf(stdout, NULL);

linklist head,pos;

int x;

head= creatLink("../data1.txt",N); /\*从data1.txt文件中读入前N个数存入带头结点的单链表head\*/

print(head); /\*当链表结点数非常大时，可选择注释本行\*/

printf("请输入要查找的整数：");

scanf("%d",&x);

clock\_t start, finish;

double total\_time;

start = clock();

pos=seqsearch(head,x);

finish = clock();

total\_time = (double) (finish - start) /CLOCKS\_PER\_SEC;

printf("%.3fs\n", total\_time);

if (pos==NULL)

printf("查找失败！\n");

else

printf("查找成功！%d\n",pos->info);

delList(head);

}

#include "ArrayIo.h"

#define N 500000 /\*数据量\*/

/\*请将本函数补充完整，并进行测试\*/

int binSearch(int a[], int n, int key)

{

int mid, left, right;

left = 0;

right = n - 1;

while (left <= right)

{

mid = (left + right) / 2;

if (a[mid] == key)

{

return mid;

} else if (a[mid] < key)

{

left = mid + 1;

} else

{

right = mid - 1;

}

}

return -1;

}

int main()

{

setbuf(stdout, NULL);

int a[N], n, x, pos;

clock\_t start, finish;

double total;

n = readData(a, N, "../data2.txt"); /\*从data2.txt文件中读入前N个数存入a中，函数返回成功读入的数据个数\*/

printf("请输入要查找的整数：");

scanf("%d", &x);

start = clock();

pos = binSearch(a, n, x);

finish = clock();

total = ((double) (finish - start)) / CLOCKS\_PER\_SEC;

printf("time is %f\n", total);

if (pos == -1)

printf("查找失败");

else

printf("a[%d]=%d\n", pos, x);

}

#include "ArrayIo.h"

#define N 10000 /\*数据量\*/

/\*请将本函数补充完整，并进行测试\*/

int binSearch(int a[],int low,int high,int key)

{

int mid;

if(low > high){

return -1;

}

else

{

mid = (low + high) /2;

if(a[mid] == key){

return mid;

}else if(a[mid] > key){

return binSearch(a, low, high -1 , key);

}else{

return binSearch(a, low+1, high, key);

}}

}

int main()

{

int a[N],n,x,pos;

n=readData(a,N,"../data2.txt"); /\*从data2.txt文件中读入前N个数存入a中，函数返回成功读入的数据个数\*/

printf("请输入要查找的整数：");

scanf("%d",&x);

clock\_t start, finish;

double total;

start = clock();

pos=binSearch(a,0,n-1,x);

finish = clock();

total = ((double) (finish - start)) / CLOCKS\_PER\_SEC;

printf("time is %f\n", total);

if (pos==-1)

printf("查找失败");

else

printf("a[%d]=%d\n",pos,x);

}

#include "Arrayio.h"

#include "bstree.h"

#define N 10

bstree creatBstree(int a[],int n)

{ /\*根据输入的结点序列，建立一棵二叉排序树，并返回根结点的地址\*/

bstree t, p, f;

int i;

t = NULL;

for(i = 0; i <= n-1; i++){

p = t;

while(p)

{

if(a[i] == p->key) break;

f = p;

p = p->key > a[i] ? p->lchild : p->rchild;

}

if(p !=NULL && p->key == a[i]) continue;

p = (bstree) malloc(sizeof(bsnode));

p->rchild = NULL;

p->lchild = NULL;

p->key = a[i];

if(!t){

t = p;

}else if(f ->key > a[i]){

f->lchild = p;

}else{

f->rchild = p;

}

}

return t;

}

int main()

{

int n,a[N];

bstree p,t;

n=readData(a,N,"../data1.txt");

output(a,n);

t=creatBstree(a,n); /\*创建二叉排序树\*/

printf("中序遍历：\n");

inorder(t); /\*中序遍历二叉排序树\*/

return 0;

}