**暨南大学本科实验报告专用纸**

课程名称 数据结构实验 成绩评定

实验项目名称 顺序、折半查找 指导教师 刘波、郭科芩

实验项目编号 8060154911 实验项目类型 综合性 实验地点 N116

学生姓名 阮炜霖 学号 2020101603

学院 信息科学技术学院 系 计算机系 专业 网络工程

实验时间 2021 年 10 月 9 日 上 午～ 10 月 9 日 上 午

**（一）实验目的和要求；**

目的：熟悉并掌握顺序、折半查找的过程及方法。

要求：任选一种高级程序语言编写源程序，并调试通过，测试正确。

**（二）实验主要内容；**

1. 在输入数组的记录中顺序查找所需的数据；
2. 对已有的有序序列进行折半查找；
3. 比较以上两种方法的查找次数。

**（三）实验原理**

对于顺序查找，我们只需要在数组中找到相应数的位置即可；对于折半查找，运用了二分思想，我们有很多方式去实现，比如C++自带的lower\_bound函数便可用折半找到相应数的位置；在这里我选择使用Splay平衡树来实现。首先将序列放入一颗BST(主要性质有左儿子值<父亲值<右儿子值)，每次插入为O(logn)。通过旋转操作来保持查找操作不会退化成线性（通过splay操作可以在logn的时间内旋出最佳的根节点），顺序查找的时间复杂度为O(n),折半查找为O(logn)。另外，Splay是一种优秀的数据结构，拥有良好的鲁棒性并且支持插入、删除、查找k大、查找x的位置、查找前驱及后继等操作。

**（四）实验步骤及调试分析；**

查阅资料习得Splay平衡树的具体写法，并且熟悉其原理，通过定义每个节点的fa,sz,val,cnt（分别为父亲节点编号、子树大小、权值、值的个数）来进行建树，并定义主要函数get\_rank以及其他相关函数：

gets(x):返回x是否为父亲的右节点

maintain(x):更新以x为根的子树大小

clear(x):清空节点x

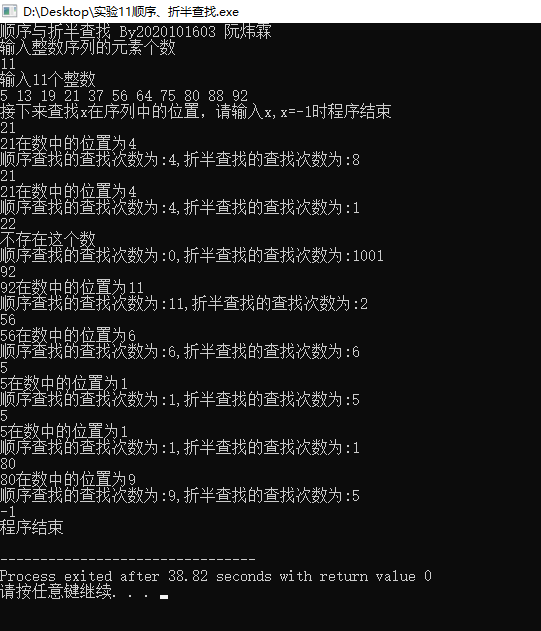
get\_rank(x):返回x在序列中的相对位置

splay(x):将x旋至根节点

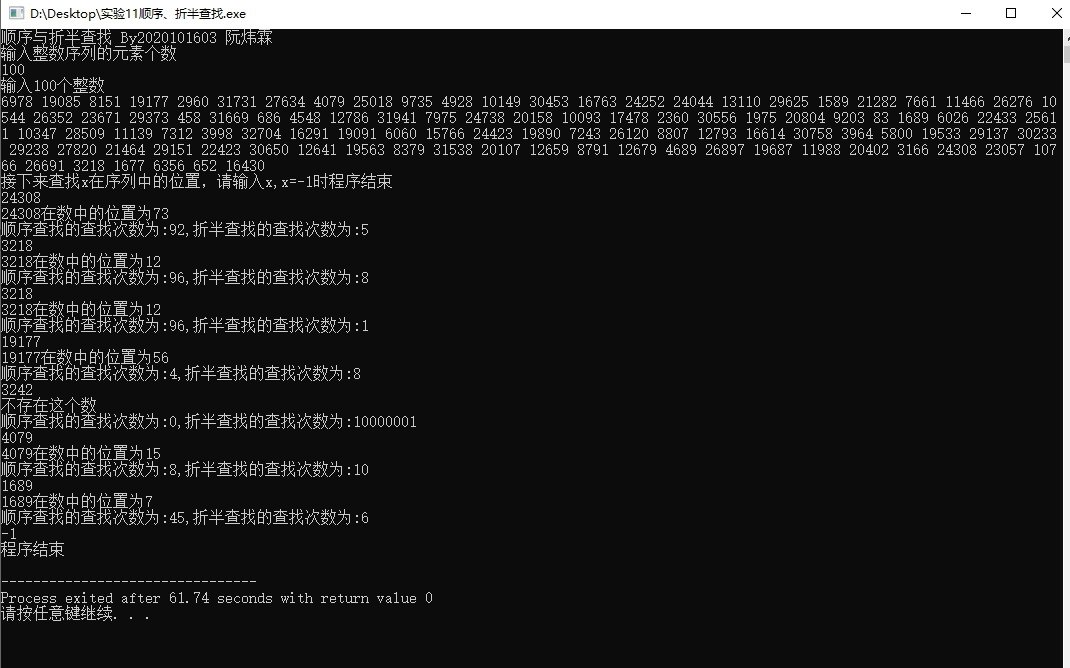
rotate(x):单次旋转操作，分为左旋和右旋。

**（五）实验结果及分析；**

通过实验结果，我们得出结论：该程序实现了利用数组对序列进行顺序查找以及利用Splay平衡树对序列进行折半查找。发现随着序列元素个数的增大，顺序查找的效率越来越低，时间复杂度为O(n)，而折半查找的效率更优，始终保持在O(logn)，虽然常数比较大，但速度上仍体现出了优越性。



(书本P216样例)



(随机数据)

**（六）附录：源程序**

#include<bits/stdc++.h>

#define get(x) (x==ch[fa[x]][1])

using namespace std;

const int maxn=1e6+5;

typedef long long ll;

int fa[maxn],sz[maxn],val[maxn],cnt[maxn];

int ch[maxn][2],rt,tot;

int n,opt,x,tim1,tim2,a[maxn];

/\*

bool get(int x){

return x==ch[fa[x]][1];

}\*/

void maintain(int x){

sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+cnt[x];

}

void clear(int x){

ch[x][0]=ch[x][1]=sz[x]=val[x]=cnt[x]=fa[x]=0;

}

void rotate(int x){

// tim1++;

int y=fa[x],z=fa[y],chk=get(x);

ch[y][chk]=ch[x][chk^1];

if(ch[x][chk^1]) fa[ch[x][chk^1]]=y;

ch[x][chk^1]=y;

fa[y]=x;

fa[x]=z;

if(z) ch[z][y==ch[z][1]]=x;

maintain(y);

}

void splay(int x){

for(int f=fa[x];f=fa[x],f;rotate(x))

if(fa[f]) rotate(get(x)==get(f)?f:x);

rt=x;

}

void insert(int k){

if(!rt){

val[++tot]=k;

cnt[tot]++;

rt=tot;

sz[tot]=1;

return;

}

int cur=rt,f=0;

while(1){

if(val[cur]==k){

cnt[cur]++;

maintain(cur);

maintain(f);

splay(cur);

break;

}

f=cur;

cur=ch[cur][val[cur]<k];

if(!cur){

val[++tot]=k;

cnt[tot]++;

fa[tot]=f;

ch[f][val[f]<k]=tot;

maintain(tot);

maintain(f);

splay(tot);

break;

}

}

}

int get\_rank(int k){

int res=0,cur=rt;

while(1){

tim1++;

if(tim1>10000000){

return 0;

}

if(k<val[cur]){

cur=ch[cur][0];

}else{

res+=sz[ch[cur][0]];

if(k==val[cur]){

splay(cur);

return res+1;

}

res+=cnt[cur];

cur=ch[cur][1];

}

}

}

int kth(int k){

int cur=rt;

while(1){

// tim1++;

if(ch[cur][0]&&k<=sz[ch[cur][0]])

cur=ch[cur][0];

else{

k-=cnt[cur]+sz[ch[cur][0]];

if(k<=0){

splay(cur);

return val[cur];

}

cur=ch[cur][1];

}

}

}

int get\_pre(){

int cur=ch[rt][0];

while(ch[cur][1]) cur=ch[cur][1];

splay(cur);

return cur;

}

int get\_nxt(){

int cur=ch[rt][1];

while(ch[cur][0]) cur=ch[cur][0];

splay(cur);

return cur;

}

void del(int k){

get\_rank(k);

if(cnt[rt]>1){

cnt[rt]--;

maintain(rt);

return;

}

if(!ch[rt][0]&&!ch[rt][1]){

clear(rt);

rt=0;

return;

}

if(!ch[rt][0]){

int cur=rt;

rt=ch[rt][1];

fa[rt]=0;

clear(cur);

return;

}

if(!ch[rt][1]){

int cur=rt;

rt=ch[rt][0];

fa[rt]=0;

clear(cur);

return;

}

int cur=rt,x=get\_pre();

fa[ch[cur][1]]=x;

ch[x][1]=ch[cur][1];

clear(cur);

maintain(rt);

}

int main(){

cout<<"顺序与折半查找 By2020101603 阮炜霖\n";

cout<<"输入整数序列的元素个数\n";

scanf("%d",&n);

cout<<"输入"<<n<<"个整数\n";

for(int i=1;i<=n;i++){

cin>>a[i]; //这是输入的数组

insert(a[i]); //在Splay平衡树中新建节点

}

cout<<"接下来查找x在序列中的位置，请输入x,x=-1时程序结束\n";

while(cin>>x){

tim1=tim2=0;

if(x==-1) break;

int rk=get\_rank(x);

if(rk) printf("%d在数中的位置为%d\n",x,rk);

else printf("不存在这个数\n");

if(tim1<=10000000) for(int i=1;i<=n;i++){

tim2++;

if(a[i]==x) break;

}

cout<<"顺序查找的查找次数为:"<<tim2<<",折半查找的查找次数为:"<<tim1<<"\n";

}

cout<<"程序结束\n";

return 0;

}

**暨南大学本科实验报告专用纸(附页)**