班级—— 2014211314

姓名—— 袁振宇

学号—— 2014211529

分工情况——个人独立完成

完成日期——2016.1.3

1. **问题描述**

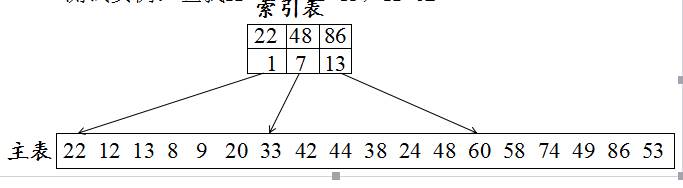
试编写利用折半查找确定记录所在块的分块查找算法。

提示：读入各记录建立主表；

按L个记录/块建立索引表；（L=6，L设置为可变值）

查找；

测试实例：查找K=48， K=13，K=62



1. **明确规定**
2. 输入形式：用户输入主表关键字个数，然后按规定分块，按有序或分段有序顺序输入关键字，最后输入要查找的关键字（以-1结束）；
3. 输出形式：输出查找结果（成功或失败）以及查找成功时该关键字在主表中的位置；
4. 程序所能达到的功能：利用折半查找确定记录所在块的分块查找。
5. **算法思想**

用折半查找确定记录所在块的分块查找。对索引表进行折半查找，在块内进行顺序查找。

1. **设计描述**
2. 索引表结构体；
3. 主函数调用MaxKey(indexItem block[], indexTable s[], int keynum, int a)得到每块的最大关键字，调用int BinSearch(indexItem block[], indexTable s[], int low, int heigh, int key, int &address, int keynum)递归实现折半查找，确定关键字所在的块，BinSearch()调用 int Search(indexItem block[], indexTable s[], int a, int key, int &address, int keynum)在块内顺序查找关键字；
4. 算法描述：

int MaxKey(indexItem block[], indexTable s[], int keynum, int a) //得到每块的最大关键字

{

int i, j, k;

int stadress = block[a].stadr;

int max = s[stadress].key;

for(i = block[a].stadr; (i < block[a].stadr+block[a].blocksize) && (i <= keynum); i++)

max = (s[i].key > max) ? s[i].key: max;

return max;

}

int Search(indexItem block[], indexTable s[], int a, int key, int &address, int keynum) //在块内顺序查找

{

int i;

for(i = block[a].stadr; (i < block[a].stadr+block[a].blocksize) && (i <= keynum); i++)

{

if(key == s[i].key)

{

address = i;

return 1;

}

}

return 0;

}

int BinSearch(indexItem block[], indexTable s[], int low, int heigh, int key, int &address, int keynum) //递归实现折半查找

{

int middr; //中间块的位置

int midmax; //中间块的最大关键字

while(low <= heigh)

{

middr = (low + heigh)/2;

midmax = block[middr].maxkey;

if(key <= midmax && middr > 1 && key > block[middr-1].maxkey)

return Search(block, s, middr, key, address, keynum);

else if(middr == 1)

{

return Search(block, s, middr, key, address, keynum);

}

else

if(key > midmax)

{

low = middr+1;

return BinSearch(block, s, low, heigh, key, address, keynum);

}

else

{

if(middr > 1)

{

heigh = middr-1;

return BinSearch(block, s, low, heigh, key, address, keynum);

}

if(middr == 1)

{

heigh = middr;

return BinSearch(block, s, low, heigh, key, address, keynum);

}

}

return 0;

}

}

**5.源程序**

#include<iostream>

#include<cstdlib>

using namespace std;

typedef struct{

int maxkey; //每块最大关键字

int stadr; //每块起始位置

int blocksize; //每块关键字个数

}indexItem; //索引项

typedef struct{

int key; //主表项

}indexTable;

int MaxKey(indexItem block[], indexTable s[], int keynum, int a) //得到每块的最大关键字

{

int i, j, k;

int stadress = block[a].stadr;

int max = s[stadress].key;

for(i = block[a].stadr; (i < block[a].stadr+block[a].blocksize) && (i <= keynum); i++)

max = (s[i].key > max) ? s[i].key: max;

return max;

}

int Search(indexItem block[], indexTable s[], int a, int key, int &address, int keynum) //在块内顺序查找

{

int i;

for(i = block[a].stadr; (i < block[a].stadr+block[a].blocksize) && (i <= keynum); i++)

{

if(key == s[i].key)

{

address = i;

return 1;

}

}

return 0;

}

int BinSearch(indexItem block[], indexTable s[], int low, int heigh, int key, int &address, int keynum) //递归实现折半查找

{

int middr; //中间块的位置

int midmax; //中间块的最大关键字

while(low <= heigh)

{

middr = (low + heigh)/2;

midmax = block[middr].maxkey;

if(key <= midmax && middr > 1 && key > block[middr-1].maxkey)

return Search(block, s, middr, key, address, keynum);

else if(middr == 1)

{

return Search(block, s, middr, key, address, keynum);

}

else

if(key > midmax)

{

low = middr+1;

return BinSearch(block, s, low, heigh, key, address, keynum);

}

else

{

if(middr > 1)

{

heigh = middr-1;

return BinSearch(block, s, low, heigh, key, address, keynum);

}

if(middr == 1)

{

heigh = middr;

return BinSearch(block, s, low, heigh, key, address, keynum);

}

}

return 0;

}

}

int main()

{

int keynum, blocknum, i, j, k=1;

int key; //待查找的关键字

int address = -1; //待查找关键字在主表中的位置

cout << "输入关键字个数:";

cin >> keynum;

indexTable s[keynum+1]; //建立主表

while(k=1)

{

cout << "输入要分块个数:";

cin >> blocknum;

if(blocknum > keynum)

cout << "\n错误!分块不合法,请重新分块!\n" << endl;

else

break;

}

indexItem block[blocknum+1]; //建立索引表

cout << "输入每块关键字个数:";

for(int num=0, i = 1; i <= blocknum; i++)

{

cin >> block[i].blocksize;

num += block[i].blocksize;

if(i == blocknum && num != keynum)

{

cout << "\n错误!请重新分配块!\n" << endl;

i = 0;

num = 0;

cout << "输入每块关键字个数:";

}

}

cout << "请按有序或分块有序方式输入关键字:" << endl; //建立主表

for(i = 1; i <= keynum; i++)

cin >> s[i].key;

cout << "主表建立成功,为:" << endl;

for(i = 1; i <= keynum; i++)

cout << s[i].key << " ";

cout << "\n分块情况为:" << endl;

for(i = 1,j = 1; j <= blocknum; j++)

{

block[j].stadr = i;

for(k = 1; k <= block[j].blocksize; k++)

{

cout << s[i].key << " ";

i++;

}

cout << "\n";

}

for(i = 1; i <= blocknum; i++)

block[i].maxkey = MaxKey(block, s, keynum, i);

cout << "每块的最大关键字为:" << endl;

for(i = 1; i <= blocknum; i++)

cout << block[i].maxkey << " ";

cout << "\n请输入要查找的元素(若不继续查找,请输入-1):" << endl;

cin >> key;

while(key != -1)

{

if(key != -1)

{

if(BinSearch(block, s, 1, blocknum, key, address, keynum) == 1)

cout << "查找成功,该关键字所在表中位置为:" << address << endl;

else

cout << "查找失败,该元素不在表中!" << endl;

}

cout << "请输入要查找的元素(若不继续查找,请输入-1):" << endl;

cin >> key;

address = -1;

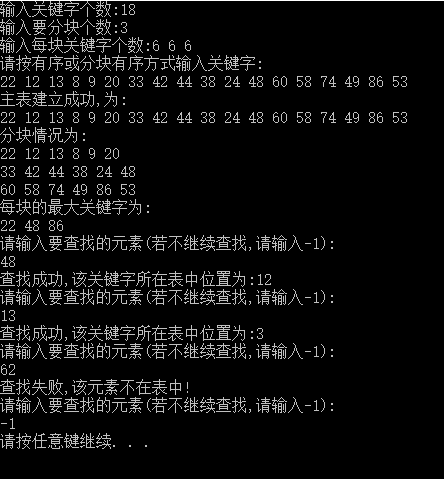
}

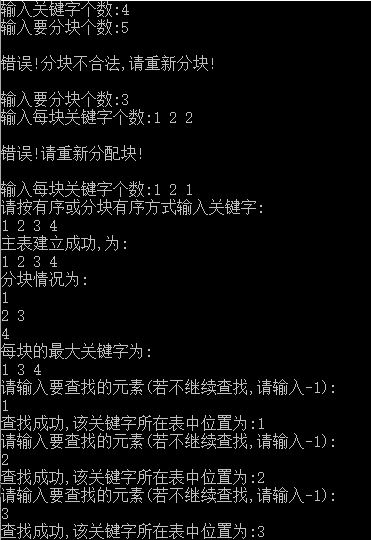
system("pause");

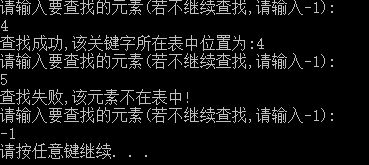
return 0;

}

**6.测试结果**







**7.用户使用说明：**

打开DEV-C++,运行程序，输入关键字时时一定要按有序或分块有序顺序输入，其余按程序提示进行操作。

**8.心得体会：**

通过本次试验，我深入体会了索引查找关键字的方法，以及递归折半查找的方法。在本程序中，我设计让用户自定义对主表进行分块，这样大大提高了分块的灵活性，在实现过程中，我进一步熟练了结构体的操作，收获颇丰。