**第五次《数据结构》编程作业**

**姓名：叶文霆**

**学号：2014211519**

**班级：2014211314**

**分工情况：全部独立完成**

**编译环境：c++ (g++编译器)**

**完成日期：12.23**

Problem 1

【问题描述】

输入 N 个已经分块有序的整数数列，其中每个块的长度为L，要求建立一个索引表，实现其静态查找功能。

输入格式：第一行共2个整形 N、L，

第二行为 N 个已经自然有序的整数。

第三行为 q，表示 询问的个数。

第四行为 Q个整数要求查找的整数x。

输出格式：共 Q行，若查找成功，输出 Yes及其下标；若查找失败，则输出 No。

【算法思想】

IndexTable[i]表示第i个块里面最大的元素的值，然后对于每一个询问 X，用二分查找找到IndexTable[i-1]<X<=IndexTable[i]。然后查找 Data[(i-1)\*L+1]~Data[i\*L]里面查找。

【设计描述】

存储结构：struct IndexTable

{

int n,l,m; //n 表示表长，l 表示每格块的长度，m 表示索引表的长度

int \*data,\*index; //分别存储数据和索引

};

处理流程及调用关系： 建立 Data，IndexTable->处理询问

算法描述：这次二分查找要查找的是IndexTable[i-1]<X<=IndexTable[i]的点，所以我对算法的返回值稍微进行了一点修改：

int BiSearch(const IndexTable &a, int num)

{

while (l<r)

{

int x= (l+r)>>1;

if (a.index[x]==num) l=r=x;

else if(a.index[x]>num) r=x-1;

else l=x+1;

}

if (a.index[l]>=num) return l; //为处理不同的边界状态

else return l+1;

}

【源程序】

#include <iostream>

#include <cmath>

using namespace std;

typedef struct it

{

int n,l,m; //n 表示表长，l 表示每格块的长度，m 表示索引表的长度

int \*data,\*index; //分别存储数据和索引

} IndexTable;

void Create\_IndexTable(IndexTable &a)

{

int tot=1;

cin>>a.n>>a.l;

a.m=ceil((float)a.n/a.l);

a.data= new int[a.n];

a.index= new int[a.m];

if (a.data==NULL || a.index==NULL) //内存溢出输出错误信息

{

cerr<<"Overflow!!"<<endl;

return;

}

for (int i=1; i<=a.m; i++)

{

int max=0;

while (tot<=i\*a.l&&tot<=a.n)

{

cin>>a.data[tot];

max=a.data[tot]>max?a.data[tot]:max;

tot++;

}

a.index[i]=max;

}

}

void Destroy\_IndexTable(IndexTable &a)

{

free(a.data);

free(a.index);

}

int BiSearch(const IndexTable &a, const int &num) //找到满足 index[i-1]<num<=index[i]

{

int l=1,r=a.m;

while (l<r)

{

int x= (l+r)>>1;

if (a.index[x]==num) l=r=x;

else if(a.index[x]>num) r=x-1;

else l=x+1;

}

if (a.index[l]>=num) return l;

else return l+1;

}

void Work(const IndexTable &a)

{

int q;

cin>>q;

for (int i=0; i<q; i++)

{

int num,pos,j;

cin>>num;

pos=BiSearch(a,num);

for (j=(pos-1)\*a.l+1; j<=pos\*a.l&&j<=a.n; j++)

if (a.data[j]==num) break;

if (j<=pos\*a.l&&j<=a.n) cout<<"Succeed! Its location is "<<j<<"."<<endl;

else cout<<"No Answer!"<<endl;

}

}

int main(int argc, char const \*argv[])

{

IndexTable MyData;

Create\_IndexTable(MyData);

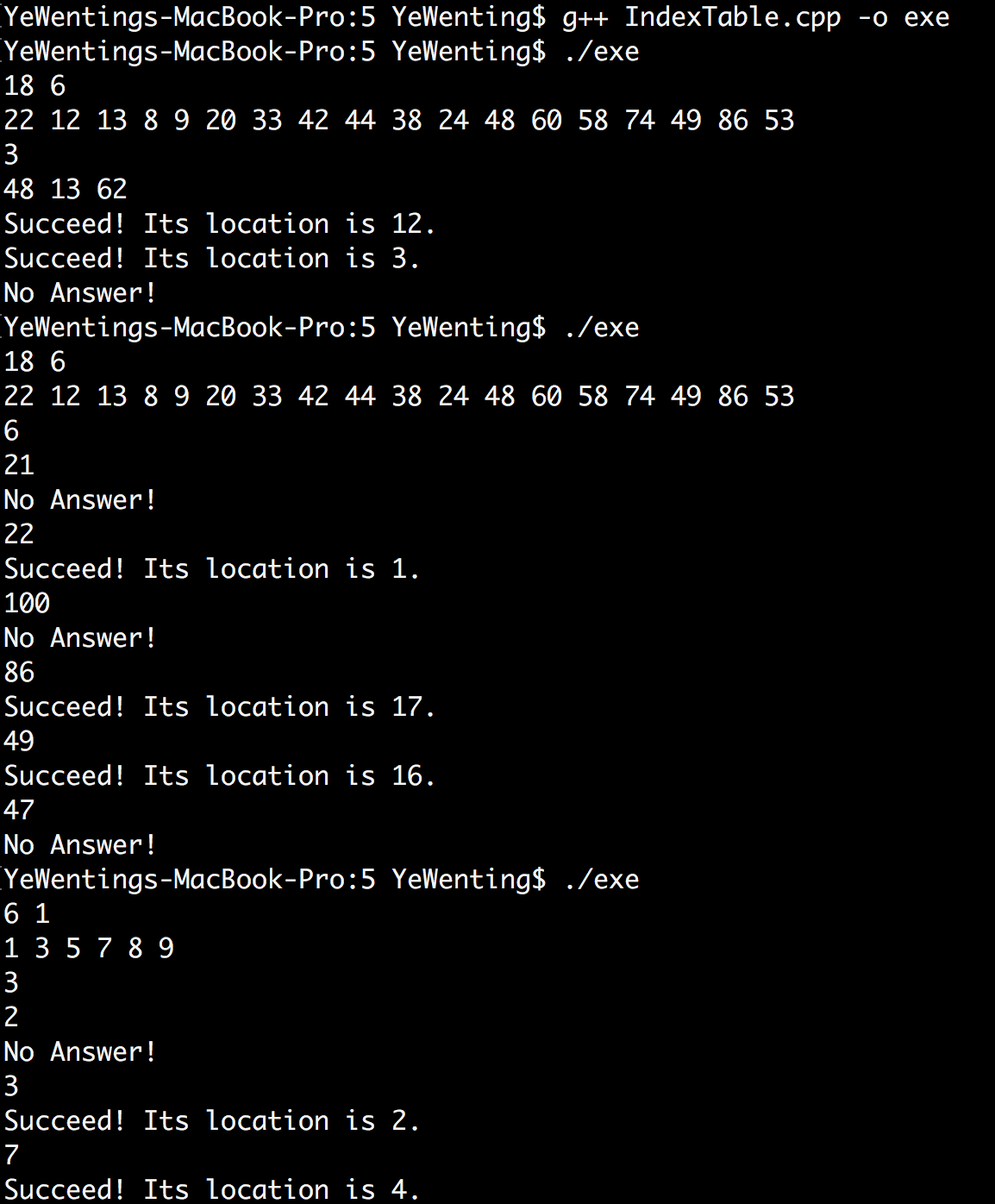
Work(MyData);

Destroy\_IndexTable(MyData);

return 0;

}

【测试说明】

****

【用户使用说明】无

【心得体会】

整道题写下来没有什么难点，就是二分查找里面每一步的条件比较值得考究，具体做的时候我是在草稿上面画了几种可能的边界状态，然后写出对应每种情况的答案，然后分类得出了我的 Bisearch()函数，第二点就是去主表去查找对应数据的时候，查找的下标范围也比较容易出错，其他的还有一点收获可能就是发现了定义结构体对于函数传参会有意想不到的效果，像这道题我定义了 IndexTable 的结构体，在函数调用的时候直接传这个变量就好了，很方便简洁。