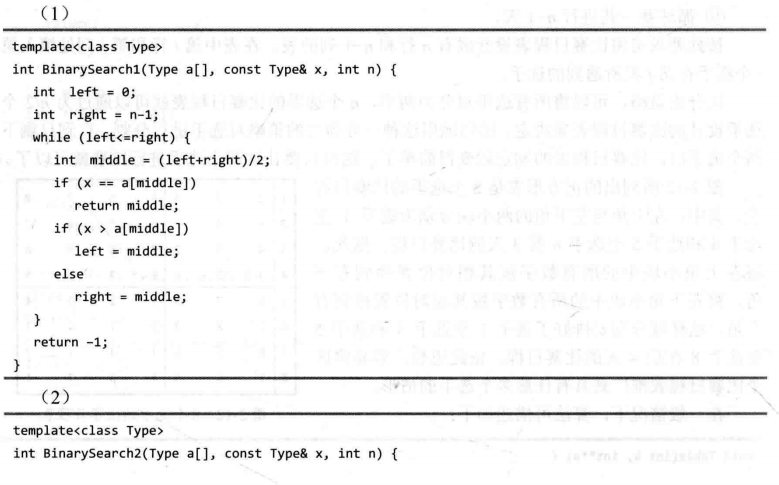
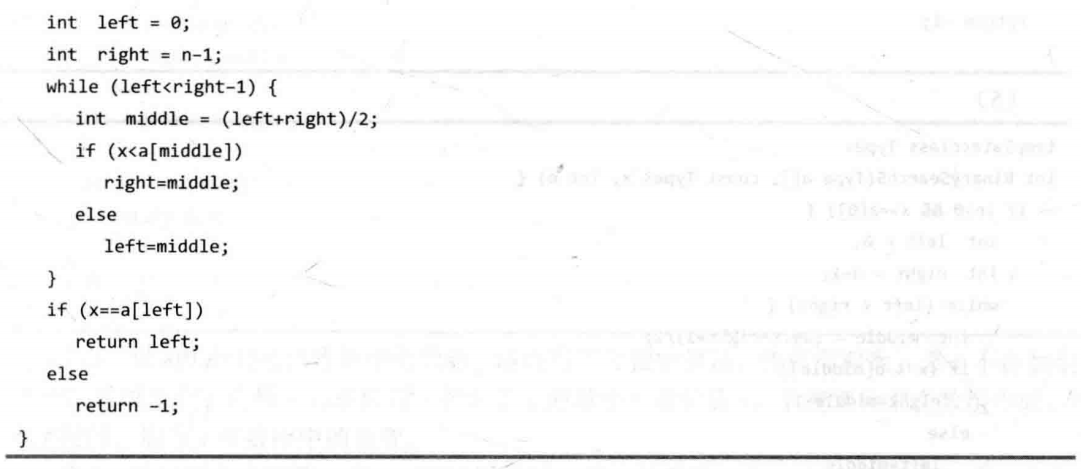
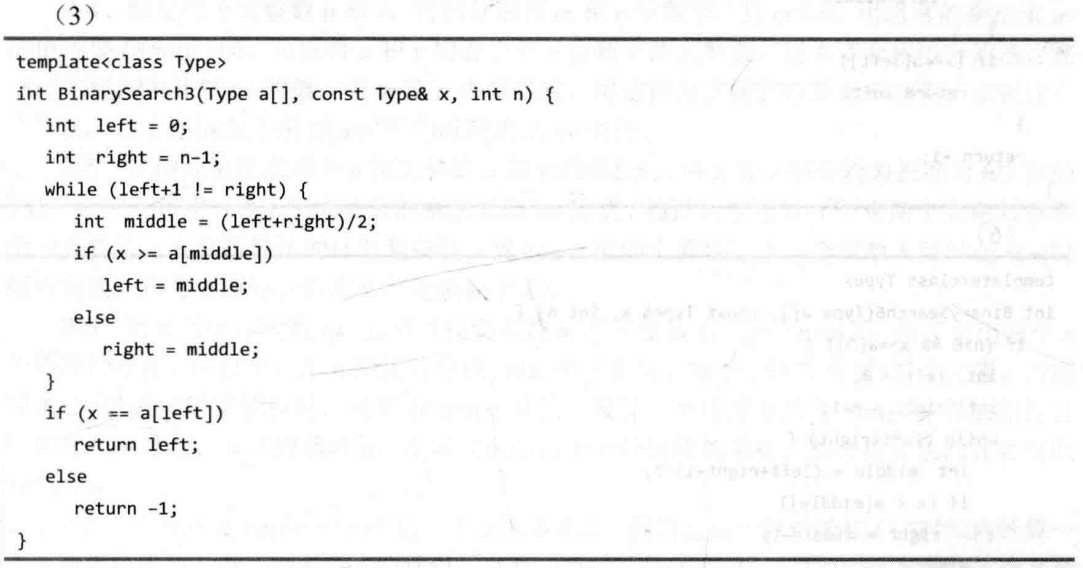
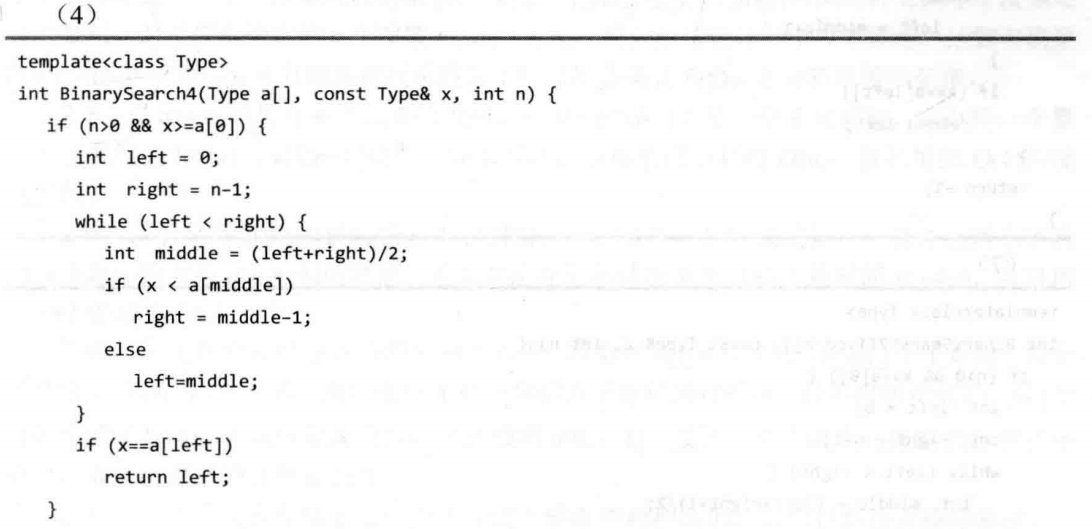
**递归与分治策略**

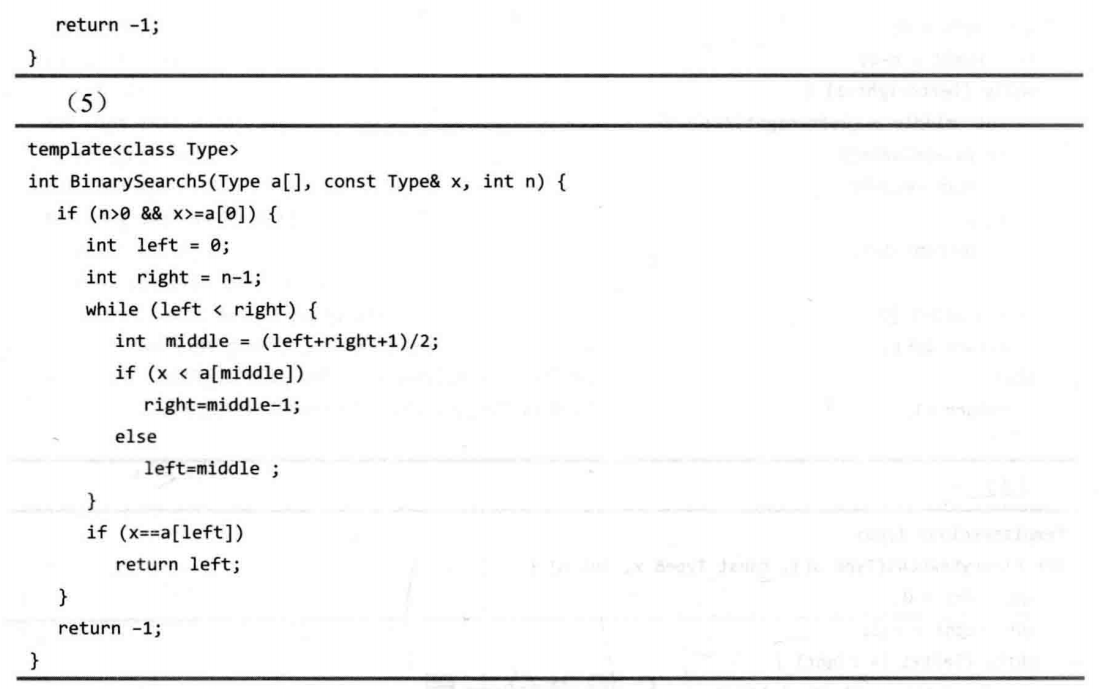
①算法分析题2-2 下面的7个算法与本章中的二分搜索算法 BinarySearch略有不同。请判断这7个算法的正确性。如果算法不正确，请说明产生错误的原因。如果算法正确，请给出算法的正确性证明。

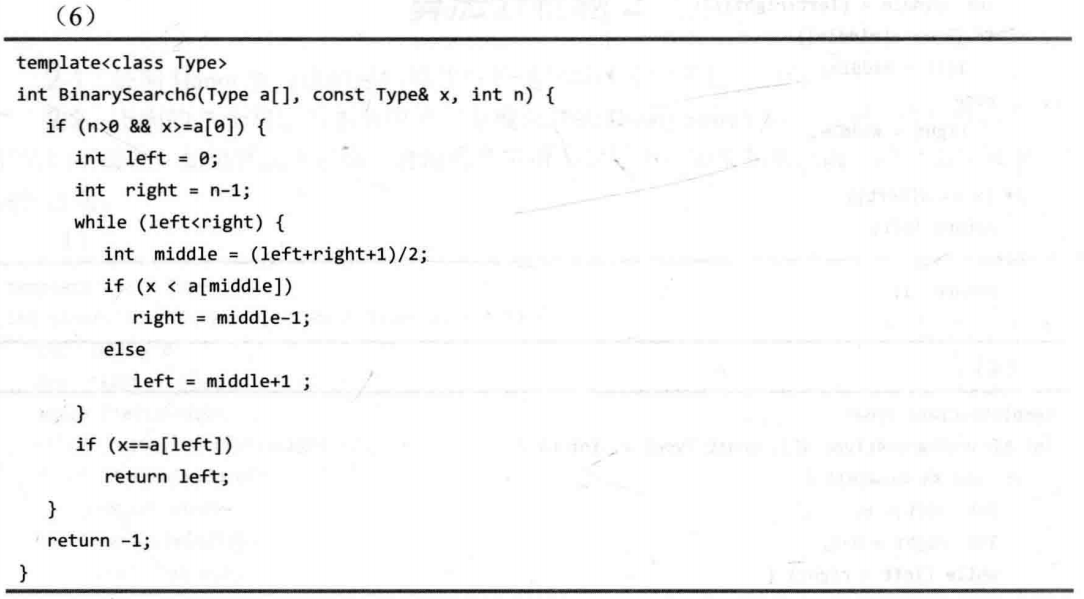


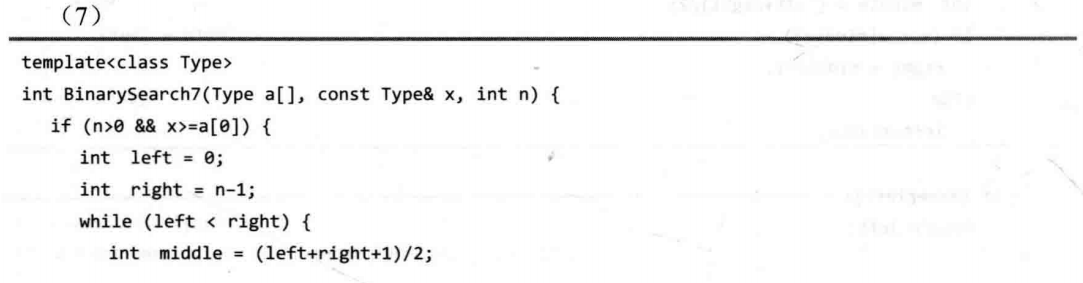


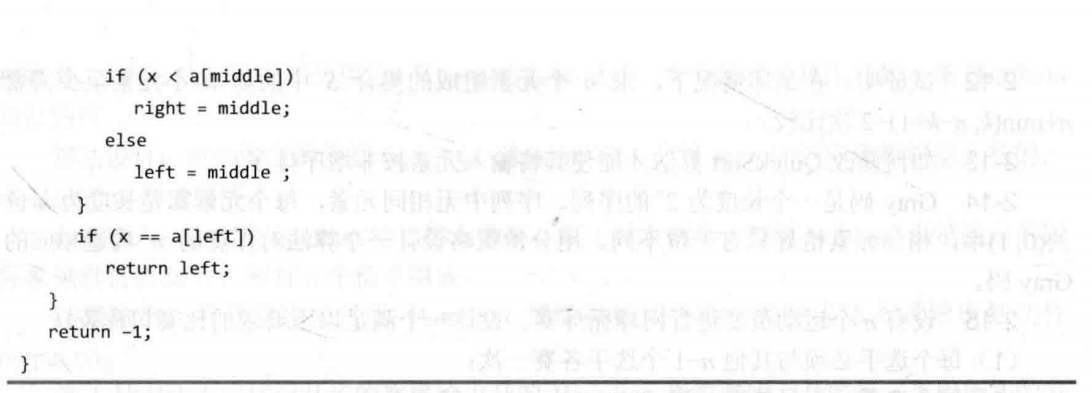












解，第一个算法与二分搜索算法BinarySearch比，当x>middle时，应该是right=middle，而不是left=middle。此时数组段左、右游标left和right的调整不正确。第二个算法数组段左、右游标left和right的调整不正确，导致当x==a[n-1]时的返回错误。第三个算法数组段左、右游标left和right的调整不正确，导致当x==a[n-1]时返回错误，第四个算法数组段左、有游标left和right的调整不正确，导致陷入死循环。

第五个算法正确，且当数组中有重复元素时，返回满足条件的最右元素。第六个算法数组段左、右游标left和right的调整不正确，导致当x==a[n-1]时返回错误。第七个算法数组段左、右游标left和right的调整不正确，导致当x=a[0]时陷入死循环。

②算法分析题2-26 试说明如何修改快速排序算法，使它在最坏情况下的计算时间为O（nlogn）

答：改变选择方式，每次选择中位数作为主元，以避免极端情况。此时递归运行时间为T(n) = 2T(n/2) + O (n) ，此时根据主方法case 2知道T(n)= O (nlogn)。

代码：

#include <iostream>

#include <time.h>

using namespace std;

const n=100;

//创建一个组成一组含有n/5个中位数的数组B

int Find(int A[n],int p,int r);//找到辅助数组B的中位数。

int PARTITION(int A[],int p,int r,int t)//t代表中位数数组B中的中位数，这里t代表为主元。

{

int i=p-1,k=0;

for (int j=p;j<=r;j++)

{

if (A[j]<t)//将比主元t大的元素交换到数组A的右边去，比主元t小的到数组A的左边。

{

i++;

swap(A[i],A[j]);

}

if (A[j]==t)//如果A[j]等于主元

{

k=j;//那么记录下主元在A中的位置。

}

}

swap(A[i+1],A[k]);//完成划分操作，主元左边的元素都小于主元，主元右边的元素都大于主元。

return i+1;

}

int SELECT(int A[],int p,int r,int i)//i表示第i小的数。

{

if (p>=r)

return A[p];

int t=Find(A,p,r);//返回的t代表辅助数组B的中位数。

int q=PARTITION(A,p,r,t);

int k=q-p+1;

if (i==k)

return A[q];

else if(i<k)

return SELECT(A,p,q-1,i);

else return SELECT(A,q+1,r,i-k);

}

int Find(int A[n],int p,int r)

{

int key=0,t=0,m=r-p+1,h=0;

if (m%5==0)

h=m/5; //如果当前数组A的大小能被5整除，那么这以5个元素为一组的m/5组数，没有余数那一组

else

h=m/5+1; //否则，应该加上含有余数的那一组。

int \*B=new int[h];

for(int j=0;j<h;j++){

B[j]=0;}

for (int k=0;k<h;k++)//5个数一组，共h组。进行插入排序。

{//经过最多h=n/5+1次循环，那么总共循环了25h=25(n/5+1)=5n+25=O(n)次

for (int j=t+1+p;j<=5+t+p&&j!=r+2;j++)//h组中每组进行插入排序。注意加上数组初始坐标p(当前数组A的初值坐标)+t(在p基础上每5个为1组)

{//运行时间分析：5个一组运行插入排序，每次插入排序需要的时间是O(n^2)=5^2=25是基于固定划分的固定常数

key=A[j-1];

int i=j-1;

while (i>t+p&&A[i-1]<key)

{

A[i]=A[i-1];

i=i-1;

}

A[i]=key;

}

t+=5;//进入下一个5个元素为一组的插入排序}

k=0;

for (int i=0;i<h&&k<h;i++)//经过最多h=n/5+1次循环(O(n))，将当前数组A中的每组的中位数依次放入到B中

{

if (i<h-1)

{

B[k]=A[2+5\*i+p];

k++;

continue;

}

if(m%5!=0)

B[k]=A[5\*i+p+(m%5)/2];

else

{

B[k]=A[2+5\*i+p];

k++;

}

}

if (h==1)

return B[0];//当辅助数组B只剩下一个数时，那么这个数就是中位数的中位数。

else

return SELECT(B,0,h-1,(h-1)/2+1);

}

void QUICKSORT(int A[],int p,int r)

{

if (p<r)

{

int t=SELECT(A,p,r,(r+p)/2-p+1);

int q=PARTITION(A,p,r,t);

QUICKSORT(A,p,q-1);

QUICKSORT(A,q+1,r);

}

}

void main()

{

int A[n]={0};

srand( (unsigned)time( NULL ) );

cout<<"排序前："<<endl;

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

{

A[i]=rand()%100;

cout<<A[i]<<"\t";

}

cout<<endl;

QUICKSORT(A,0,n-1);

cout<<"排序后："<<endl;

for (i=0;i<n;i++)

{

cout<<A[i]<<"\t";

}

cout<<endl;

}

③算法实现题2-2 问题描述：给定含有n个元素的多重集合S，每个元素在S中出现的次数称为该元素的重数。多重集S中重数最大的元素称为众数。

例如，S={1, 2, 2, 2, 3, 5}。

多重集S的众数是2，其重数为3。

编程任务：对于给定的由n个自然数组成的多重集S，编程计算S的众数及其重数。

解：

#include "stdio.h"

#include "stdlib.h"

# define N 100

void Split(int a[],int n,int &l,int &r)

{//以中间数字为界，确定左右界

int mid = n/2;

for(l = 0;l <= mid;++l)//找左界

if(a[l] == a[mid])

break;//此时l为左界

for(r = mid + 1;r < n;++r)//找右界

if(a[r] !=a[mid])

break;//此时r为右界

}

void getMaxNum(int &num,int &maxnum,int a[],int n)

{//求众数和重数

int l,r,s;

int mid = n/2;

Split(a,n,l,r);

s = r - l;

if(s > maxnum)

{//如果中间数字的个数大于现在的重数，则更新

num = a[mid];

maxnum = s;

}

if(s == maxnum)//如果出现多个众数，输出最小的那个

if(num > a[mid])

{

num = a[mid];

maxnum = s;

}

if(l+1 > maxnum)

//如果左边的个数>maxnum,则搜索左边

getMaxNum(num,maxnum,a,l+1);

if(n - r > maxnum)

//如果右边的个数>maxnum,则搜索右边

getMaxNum(num,maxnum,a+r,n-r);

}

int main()

{

int n,a[N]; // i为行数

Int i = 0;

int num = 0;//num为众数

int maxnum = 0;//maxnum为重数

FILE \*fp;

fp = fopen(“input.txt”,”r”);

while(!feof(fp)){

a[i] = fscanf(fp,"%d",&n);//输入数据

if( i = 0){

n = a[i];

i++;

}

else{

a[N] = a[i];

i++;

}

}

fw = fopen(“output.txt”,”w”);

getMaxNum(num,maxnum,a,n);

fprintf(fw,”%d\n%d”,num,maxnum);

return 0;

}