**算法训练 区间k大数查询**

时间限制：1.0s   内存限制：256.0MB

问题描述

给定一个序列，每次询问序列中第l个数到第r个数中第K大的数是哪个。

输入格式

第一行包含一个数n，表示序列长度。

第二行包含n个正整数，表示给定的序列。

第三个包含一个正整数m，表示询问个数。

接下来m行，每行三个数l,r,K，表示询问序列从左往右第l个数到第r个数中，从大往小第K大的数是哪个。序列元素从1开始标号。

输出格式

总共输出m行，每行一个数，表示询问的答案。

样例输入

5  
1 2 3 4 5  
2  
1 5 2  
2 3 2

样例输出

4  
2

数据规模与约定

对于30%的数据，n,m<=100；

对于100%的数据，n,m<=1000；

保证k<=(r-l+1)，序列中的数<=106。

锦囊1

使用排序算法。

锦囊2

对于每次询问，将询问的区间取出存储到另一个数组里面，对新的数组进行排序并输出其中第K大的。

本题的C++参考代码如下：

#include<stdio.h>

#include<algorithm>

using namespace std;

#define N 10005

int n,m;

int a[N],tree[15][N],sum[15][N];

bool cmp(int x,int y)

{

return x>y;

}

void build(int x,int L,int R)

{

if(L==R) return;

int i;

int mid=(L+R)>>1;

int lp=L,rp=mid+1,l=mid-L+1;

for(i=L;i<=R;i++)

if(tree[x][i]>a[mid])

l--;

for(i=L;i<=R;i++)

{

sum[x][i]=i==L?0:sum[x][i-1];

if(a[mid]==tree[x][i] && l)

{

l--;

tree[x+1][lp++]=tree[x][i];

sum[x][i]++;

}

else

if(tree[x][i]>a[mid])

{

tree[x+1][lp++]=tree[x][i];

sum[x][i]++;

}

else

tree[x+1][rp++]=tree[x][i];

}

build(x+1,L,mid);

build(x+1,mid+1,R);

}

int query(int x,int L,int R,int l,int r,int k)

{

if(l==r) return tree[x][l];

int mid=(L+R)>>1;

int t,tt;

t=l==L?0:sum[x][l-1];

tt=sum[x][r]-t;

if(k<=tt)

return query(x+1,L,mid,L+t,L+t+tt-1,k);

else

return query(x+1,mid+1,R,mid-L+1+l-t,mid-L+1+r-t-tt,k-tt);

}

int main()

{

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

tree[0][i]=a[i];

}

sort(a+1,a+1+n,cmp);

build(0,1,n);

scanf("%d",&m);

int l,r,k;

while(m--)

{

scanf("%d%d%d",&l,&r,&k);

printf("%d\n",query(0,1,n,l,r,k));

}

return 0;

}

本题的C参考代码如下：#include <stdio.h>

#include <stdlib.h>

int Split(int \*data,int pre,int rear)

{

int value=data[pre];

while(pre<rear)

{

while(data[rear]>=value && pre<rear) rear--;

data[pre]=data[rear];

while(data[pre]<value && pre<rear) pre++;

data[rear]=data[pre];

}

data[pre]=value;

return pre;

}

//快速排序

void QuickSort(int \*data,int pre,int rear,int k)

{

if(pre<=rear)

{

int mid=Split(data,pre,rear);

if(mid==k)

{

printf("%d\n",data[mid]);

}

else if(mid>k)

{

QuickSort(data,pre,mid-1,k);

}

else

{

QuickSort(data,mid+1,rear,k);

}

}

}

void Copy(int \*data,int n,int \*temp)

{

int i;

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

{

temp[i]=data[i];

}

}

int main()

{

int i;

int n;

int m;

int \*data;

scanf("%d",&n);

data=(int \*)malloc(sizeof(int)\*n);

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

{

scanf("%d",&data[i]);

}

scanf("%d",&m);

while(m)

{

int pre;

int rear;

int k;

int \*temp=(int \*)malloc(sizeof(int)\*n);

scanf("%d%d%d",&pre,&rear,&k);

Copy(data,n,temp);

QuickSort(temp,pre-1,rear-1,rear-k);

m--;

}

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedInputStream;

import java.io.IOException;

import java.util.Arrays;

public class Main

{

private static BufferedInputStream in = new BufferedInputStream(System.in);

public static void main(String[] args) throws IOException

{

int[] nums = new int[readInt()];

for(int i=0; i<nums.length; i++)

{

nums[i] = readInt();

}

for(int i=readInt(); i>0; i--)

{

int a = readInt();

int b = readInt();

int c = readInt();

int[] tn = new int[b-a+1];

for(int j=0; j<tn.length; j++)

{

tn[j] = nums[a-1+j];

}

Arrays.sort(tn);

System.out.println(tn[tn.length-c]);

}

}

private static int readInt() throws IOException

{

int i,sum=0;

while(((i=in.read())&48) != 48 || i>57);

for(;(i&56) == 48 || (i&62) == 56; i=in.read())

sum = sum\*10 + (i&15);

return sum;

}

}