**P1991 -- AKF的故事之找DKF的数**

时间限制：3000MS      内存限制：131072KB      通过/提交人数：31/69

状态：Accepted      标签：    分治   数据结构-树形-主席树   无

**题目描述**

#define D 第

#define F 小  (你可以将就着把它理解成Few)

给出一个长度为N的序列A1,A2,A3,...,AN，其中每项都是小于等于10^9+7的自然数。

现在有M个询问，每个询问都是Ai...Aj中DKF的数等于多少。

**输入格式**

第一行两个正整数N，M。

第二行N个数，表示序列A1,A2,...,AN。

紧着的M行，每行三个正整数i,j,k(k≤j-i+1)，表示 询问Ai...Aj中第k小的数等于多少。

**输出格式**

共输出M行，第i行输出第i个询问的答案。

**输入样例**

4 3

4 1 2 3

1 3 1

2 4 3

1 4 4

**输出样例**

1

3

4

**数据规模与约定**

对于20%的数据，M ≤ 500；

对于40%的数据，N ≤ 5000；

对于100%的数据，N ≤ 100000，M ≤ 3000.

#include<cstdio>

#include<algorithm>

#define MaxN 100010

using namespace std;

struct TreeNode{int size,lson,rson,l,r;}Tree[MaxN\*20];

int N,M,Total,A[MaxN],SortA[MaxN],Root[MaxN];

int Build\_Tree(int l,int r)

{

int now=++Total,mid=(l+r)>>1;

Tree[now].l=l;Tree[now].r=r;

if (l!=r)

{

Tree[now].lson=Build\_Tree(l,mid);

Tree[now].rson=Build\_Tree(mid+1,r);

}

return now;

}

int NewNode(int OldeNode)

{

++Total;

Tree[Total].l=Tree[OldeNode].l;

Tree[Total].r=Tree[OldeNode].r;

Tree[Total].size=Tree[OldeNode].size+1;

return Total;

}

int Modify(int OldNode,int val)

{

int now=NewNode(OldNode);

int mid=(Tree[now].l+Tree[now].r)>>1;

if (Tree[now].l==Tree[now].r) return now;

if (val<=mid)

{

Tree[now].lson=Modify(Tree[OldNode].lson,val);

Tree[now].rson=Tree[OldNode].rson;

}

else

{

Tree[now].lson=Tree[OldNode].lson;

Tree[now].rson=Modify(Tree[OldNode].rson,val);

}

return now;

}

int Query(int VerL,int VerR,int k)

{

if (Tree[VerL].l==Tree[VerL].r) return Tree[VerL].l;

int DeltaL=Tree[Tree[VerR].lson].size-Tree[Tree[VerL].lson].size;

//DeltaL为 lnow~rnow段 左子树对应区间增加的数的个数

if (DeltaL>=k) return Query(Tree[VerL].lson,Tree[VerR].lson,k);

else return Query(Tree[VerL].rson,Tree[VerR].rson,k-DeltaL);

}

int main()

{

scanf("%d%d",&N,&M);

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

{

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

SortA[i]=A[i];

}

sort(SortA+1,SortA+1+N);

int Diff\_Val=unique(SortA+1,SortA+1+N)-(SortA+1);

Root[0]=Build\_Tree(1,Diff\_Val);

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

{

int pos=lower\_bound(SortA+1,SortA+Diff\_Val+1,A[i])-SortA;

Root[i]=Modify(Root[i-1],pos);

}

int AskL,AskR,AskK;

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

{

scanf("%d%d%d",&AskL,&AskR,&AskK);

printf("%d\n",SortA[ Query(Root[AskL-1],Root[AskR],AskK) ]);

}

return 0;

}

主席树核心：

构造权值线段树（ie。以A[i] 的值为下标）

从1—>N 初始化 每次在A[i] 为下标的地方加上1

通常：二分答案(mid) 对于每次二分出来的结果mid

用两个前缀和对减判断 ( 如本题中

let X = A1~A[i-1]中，数值在区间[1,mid-1] 内的个数

let Y = A1~Aj中，数值在区间[1,mid-1]内的个数

则mid是区间第(x-y+1)大的数 )