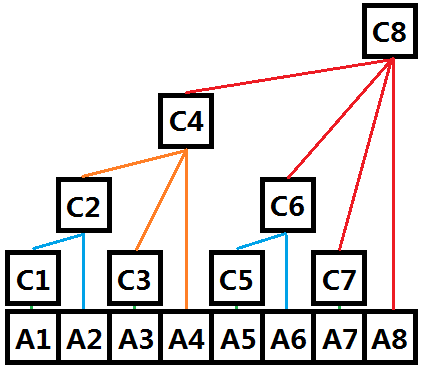
0205\_vjudge hdu-3333 Turing tree 树状数组+离线查询 解法：

emmm，这题太坑爹了，开始是想用RMQ写，然而...唉，放弃了，不过发现了树状数组这个神奇的数据结构,开始尝试了朴素的查询方法，即每提供一个区间就查询一次，不出所料，就算用了树状数组也T了...最后还是在网上搜了题解，发现了另一种骚操作--离线查询，这里离线查询的基础是在数组的右端出现之前出现过的数时，应该尽可能删除左端的数，这样只要上界接近，就不需要管左端的点，避免了“需要把删除的数补回来”。

首先把查询区间根据右端点排序，然后用一个map记录下每个数字最右出现的位置，再根据树状数组查询区间和。

讲解的比较透彻的树状数组：<http://www.cppblog.com/menjitianya/archive/2015/11/02/212171.html>



上图A为普通数组，C为构建的树状数组，Ci的值为其子节点与Ai的和，其中C的构建方法为i转换成二进制数，末尾的0的个数为k，I=i+2^k，I则为i的父节点位置。

**1.**求i末尾0的2^k的O(1)实现：x & (-x) (其中 x >= 0)，

首先进行&运算，我们需要将x和-x都转化成补码，然后再看&之后会发生什么，我们假设 x 的二进制表示的末尾是连续的 k 个 0，令x的二进制表示为 X0X1X2…Xn-2Xn-1,则{Xi = 0 | n-k <= i < n}，这里的X0表示符号位。

x 的补码就是由三部分组成：(0)(X1X2…Xn-k-1)(k个0)其中Xn-k-1为1，因为末尾是k个0，如果它为0，那就变成k+1个0了。

-x的补码也是由三部分组成：(1)(Y1Y2…Yn-k-1)(k个0)其中Yn-k-1为1，其它的Xi和Yi相加为1，想想补码是怎么计算的就明白了。

那么 x & (-x) 也就显而易见了，由两部分组成 (1)(k个0)，表示成十进制为 2k 啦。

由于&的优先级低于-，所以代码可以这样写：

int lowbit(int x){

return x&-x;

}

**2.**更新节点：把Ai的值累加到Clowbit(i)上

int add(int cnt,int v){

for(int i=cnt;i<=n;i+=lowbit(i))

c[i]+=v;

}

**3.**求和：

sum(i) = A[1] + A[2] + ... + A[i]

= A[1] + A[2] + A[i-2^k] + A[i-2^k+1] + ... + A[i]

= A[1] + A[2] + A[i-2^k] + C[i]

= sum(i - 2^k) + C[i]

= sum( i - lowbit(i) ) + C[i]

long long sum(int x){

return x?c[x]+sum(x-lowbit(x)):0;

}

树状数组的PUIQ模型基本操作如上，是简单快捷性价比极高的数据结构！

A题代码如下：

#include <map>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int a[30005],n;

long long c[30005],tmp[100005];

map<int,int> mark;

struct \_query{

int l,r,id;

}q[100005];

bool cmp(\_query a,\_query b){

return a.r<b.r;

}

int lowbit(int x){

return x&-x;

}

int add(int cnt,int v){

for(int i=cnt;i<=n;i+=lowbit(i))

c[i]+=v;

}

long long sum(int x){

return x?c[x]+sum(x-lowbit(x)):0;

}

int main(){

int t;

scanf("%d",&t);

while(t--){

mark.clear();

memset(c,0,sizeof(c));

scanf("%d",&n);

for(int i=1;i<=n;i++) scanf("%d",&a[i]);

int m;

scanf("%d",&m);

for(int i=0;i<m;i++){

scanf("%d%d",&q[i].l,&q[i].r);

q[i].id=i;

}

sort(q,q+m,cmp);

int id=1;

for(int i=0;i<m;i++){

while(id<=q[i].r){

if(mark[a[id]]!=0){

add(mark[a[id]],-a[id]);

add(id,a[id]);

mark[a[id]]=id;

}

else{

add(id,a[id]);

mark[a[id]]=id;

}

id++;

}

tmp[q[i].id]=sum(q[i].r)-sum(q[i].l-1);

}

for(int i=0;i<m;i++) printf("%lld\n",tmp[i]);

}

}