[**左式堆（左偏树）**](http://www.cppblog.com/notonlysuccess/archive/2009/03/18/77044.html)

<http://acm.hdu.edu.cn/showproblem.php?pid=1512>

#include<stdio.h>  
#define MAX 100001  
  
struct Left\_Heap{  
    int left,right;  
    int father;  
    int pow;  
    int deep;  
}hh[MAX];  
  
int find(int a)  
{  
    while(a!=hh[a].father)  
        a = hh[a].father;  
    return a;  
}  
int unite(int x,int y)//返回的是树的根节点  
{  
    if(!x)            //x是空树  
        return y;    //返回的就是y树的根节点  
    if(!y)  
        return x;  
    if(hh[x].pow < hh[y].pow)            //以x树主树，y插入  
        x^=y^=x^=y;  
    hh[x].right = unite(hh[x].right,y);    //合并y和x的右子树  
    hh[ hh[x].right ].father = x;        //x的右子树的父亲是x(这不是废话嘛)  
    if(hh[ hh[x].left ].deep < hh[ hh[x].right ].deep)//保证左子深度要比右子树大  
        hh[x].left ^= hh[x].right ^= hh[x].left ^= hh[x].right;  
    hh[x].deep = hh[ hh[x].right ].deep + 1;  
    return x;  
}  
int maxpow(int k)//返回的是树的根节点  
{  
    int l,r;  
    hh[k].pow >>= 1;  
    l = hh[k].left;  
    r = hh[k].right;//节点单独拿出来  
    hh[l].father = l;  
    hh[r].father = r;//左右儿子分别成树  
    hh[k].left = hh[k].right = hh[k].deep = 0;//变成没有左右儿子  
    l = unite(l,r);//合并原来的左右儿子  
    k = unite(l,k);//把这个节点加进合并出来的树  
    return k;  
}  
int main()  
{  
    int n,i,m,a,b,x,y,k;  
    while(scanf("%d",&n)==1)  
    {  
        for(i=1;i<=n;i++)  
        {  
            scanf("%d",&hh[i].pow);  
            hh[i].father = i;  
            hh[i].left = hh[i].right = hh[i].deep = 0;  
        }  
        scanf("%d",&m);  
        while(m--)  
        {  
            scanf("%d%d",&a,&b);  
            x = find(a);//找根节点  
            y = find(b);//找根节点  
            if(x==y)  
                puts("-1");  
            else  
            {  
                x = maxpow(x);//处理x的根节点，并返回处理后的根节点  
                y = maxpow(y);  
                k = unite(x,y);//合并两颗树  
                printf("%d\n",hh[k].pow);  
            }  
        }  
    }  
    return 0;  
}

合并的时候的步骤  
0.其中一颗树为空的话返回另外一个  
1.将x确定为主树(交换xy)  
2.合并x的右儿子和y  
3.合并出来的根节点的father指向x  
4.保证左儿子深度大于右儿子   
  
还有一道：  
<http://acm.hdu.edu.cn/showproblem.php?pid=1434>