# **[Monkey King（左偏树 可并堆）](https://www.cnblogs.com/caijiaming/p/10447298.html)**

　　我们知道如果要我们给一个序列排序，按照某种大小顺序关系，我们很容易想到优先队列，的确很方便，但是优先队列也有解决不了的问题，当题目要求你把两个优先队列合并的时候，这就实现不了了

优先队列只有插入 删除 取数的操作，但是却没有合并两个优先队列的操作。 这也是它的局限所在。

　　本次要介绍的左偏树拥有优先队列的所有功能，同时它还可以合并操作。  树的复杂度都比较低，一般log（n）就够了，左偏树也是如此，左偏树如果一个个结点暴力插入复杂度最大为nlog（n）

还有一种仿照二叉树的算法，这里不做介绍。  下面仔细说一下什么是左偏树：

　　H ← Merge(H1,H2)：

　　Merge( ) 构造并返回一个包含H1和H2所有元素的新堆H。

　　左偏树的定义：左偏树(Leftist Tree)是一种可并堆的实现。左偏树是一棵二叉树，它的节点除了和二叉树的节点一样具有左右子树指针( left, right )外，

还有两个属性：键值和距离(dist)。键值上面已经说过，是用于比较节点的大小。距离则是如下定义的：

节点 i 称为外节点(external node)，当且仅当节点 i 的左子树或右子树为空( left(i) = NULL 或 right(i) = NULL )；

节点 i 的距离( dist( i ) )是节点 i 到它的后代中，最近的外节点所经过的边数。特别的，如果节点 i 本身是外节点，则它的距离为 0；

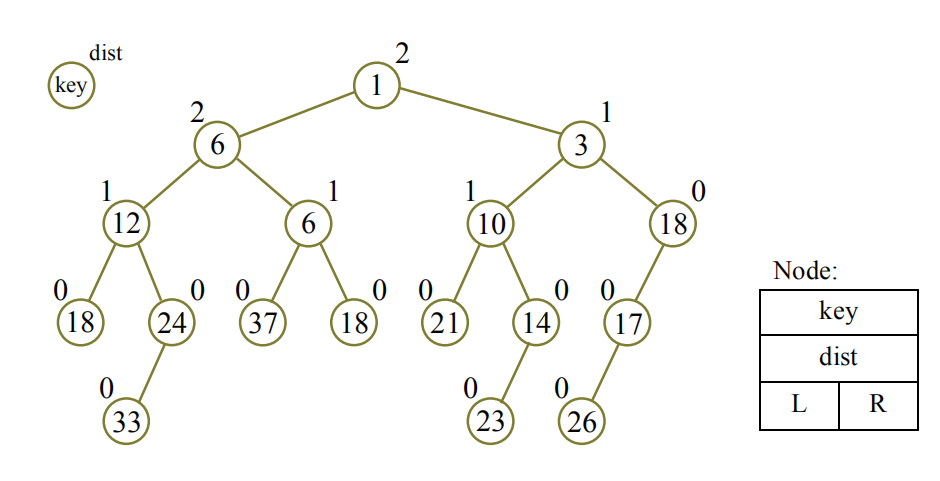
而空节点的距离规定为-1 (dist(NULL) = -1)。在本文中，有时也提到一棵左偏树的距离，这指的是该树根节点的距离。

左偏树满足下面两条基本性质：

[性质 1] 节点的键值小于或等于它的左右子节点的键值。

即 key(i)≤key(parent(i)) 这条性质又叫堆性质。符合该性质的树是堆有序

的(Heap-Ordered)。有了性质 1，我们可以知道左偏树的根节点是整棵树的最小 节点(也可以使得它最大)，于是我们可以在 O(1) 的时间内完成取最小节点操作。   
  
  
[性质 2] 节点的左子节点的距离不小于右子节点的距离。   
即 dist(left(i))≥dist(right(i)) 这条性质称为左偏性质。性质 2 是为了使我们可以以更小的代价在优先队列的其它两个基本操作（插入节点、删除最小节点）  
  
进行后维持堆性质。在后面我们就会看到它的作用。  
  
这两条性质是对每一个节点而言的，因此可以简单地从中得出，左偏树的左右子树都是左偏树。  
  
由这两条性质，我们可以得出左偏树的定义：左偏树是具有左偏性质的堆有序二叉树。  
  
下图是一棵左偏树：

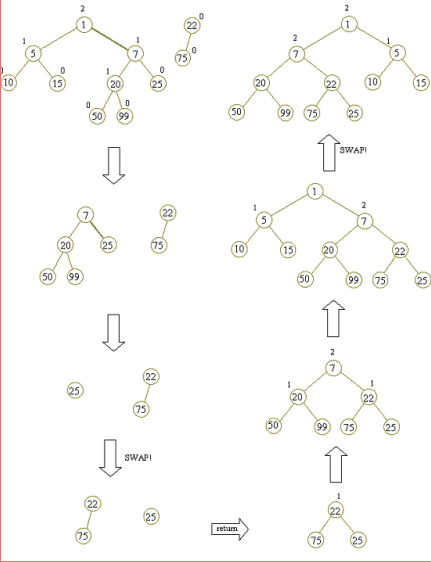


2.3 左偏树的性质   
在前面一节中，本文已经介绍了左偏树的两个基本性质，下面本文将介绍左偏树的另外两个性质。  
  
我们知道，一个节点必须经由它的子节点才能到达外节点。由于性质 2，一个节点的距离实际上就是这个节点一直沿它的右边到达一个外节点所经过的边数，也就是说，我们有  
[性质 3] 节点的距离等于它的右子节点的距离加 1。  
即 dist( i ) = dist( right( i ) ) + 1 外节点的距离为 0，由于性质 2，它的右子节点必为空节点。为了满足性质 3，故前面规定空节点的距离为-1。  
  
我们的印象中，平衡树是具有非常小的深度的，这也意味着到达任何一个节点所经过的边数很少。左偏树并不是为了快速访问所有的节点而设计的，它的目的是快速访问最小节点以及在对树修改后快速的恢复堆性质。从图中我们可以看到它并不平衡，由于性质 2 的缘故，它的结构偏向左侧，不过距离的概念和树的深度并不同，左偏树并不意味着左子树的节点数或是深度一定大于右子树。  
  
下面我们来讨论左偏树的距离和节点数的关系。  
[引理 1] 若左偏树的距离为一定值，则节点数最少的左偏树是完全二叉树。  
证明：由性质 2 可知，当且仅当对于一棵左偏树中的每个节点 i，都有dist(left(i)) = dist(right(i)) 时，该左偏树的节点数最少。显然具有这样性质的二叉树是完全二叉树。  
  
  
[定理 1] 若一棵左偏树的距离为k，则这棵左偏树至少有 2k+1-1 个节点。  
证明：由引理 1 可知，当这样的左偏树节点数最少的时候，是一棵完全二叉  
树。距离为k的完全二叉树高度也为k，节点数为 2k+1-1，所以距离为k的左偏树  
至少有 2k＋1  
-1 个节点。  
作为定理 1 的推论，我们有：  
[性质 4] 一棵 N 个节点的左偏树距离最多为 ⎣log(N+1)⎦ -1。   
证明：设一棵N个节点的左偏树距离为k，由定理 1 可知，N ≥ 2k+1-1，因此k   
≤ ⎣log(N+1)⎦ -1。  
有了上面的 4 个性质，我们可以开始讨论左偏树的操作了

1、左偏树的合并

C ← Merge(A,B)

下图是一个合并过程的示例：



下面是合并的代码：

int Merge(int x,int y)//两棵树合并{

if(!x) return y;//找到插入的地方了

if(!y) return x;

if(v[x]<v[y]) swap(x,y);//使得根节点最大

r[x]=Merge(r[x],y);//递归合并右子树和y

fa[r[x]]=x;//更新右子树的根

if(d[l[x]]<d[r[x]]) swap(l[x],r[x]);

d[x]=d[r[x]]+1;

return x;//新的根

}

2、插入新节点：

int Insert(int x, int y)

{

return Merge(x, Init(y));

}

3、删除最小节点：

int Pop(int x)

{

int L=l[x];//取出左子树

int R=r[x];//取出右子树

fa[L]=L;//根变为自己

fa[R]=R;

v[x]/=2;//取根节点 值除二

r[x]=l[x]=d[x]=0;

return Merge(L,R);//左右子树合并 返回值为左右子树合并后新的根

}

下面看一道例题：

题目链接：http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemId=1389

Monkey King

Time Limit: 10 Seconds      Memory Limit: 32768 KB

Once in a forest, there lived N aggressive monkeys. At the beginning, they each does things in its own way and none of them knows each other. But monkeys can't avoid quarrelling, and it only happens between two monkeys who does not know each other. And when it happens, both the two monkeys will invite the strongest friend of them, and duel. Of course, after the duel, the two monkeys and all of their friends knows each other, and the quarrel above will no longer happens between these monkeys even if they have ever conflicted.

Assume that every money has a strongness value, which will be reduced to only half of the original after a duel(that is, 10 will be reduced to 5 and 5 will be reduced to 2).

And we also assume that every monkey knows himself. That is, when he is the strongest one in all of his friends, he himself will go to duel.

Input

There are several test cases, and each case consists of two parts.

First part: The first line contains an integer N(N<=100,000), which indicates the number of monkeys. And then N lines follows. There is one number on each line, indicating the strongness value of ith monkey(<=32768).

Second part: The first line contains an integer M(M<=100,000), which indicates there are M conflicts happened. And then M lines follows, each line of which contains two integers x and y, indicating that there is a conflict between the Xth monkey and Yth.

**Output**

For each of the conflict, output -1 if the two monkeys know each other, otherwise output the strongness value of the strongest monkey in all friends of them after the duel.

Sample Input

5  
20  
16  
10  
10  
4  
5  
2 3  
3 4  
3 5  
4 5  
1 5

**Sample Output**

8  
5  
5  
-1  
10

 题目大意：

题意：  N(N<=10^5)只猴子，初始每只猴子为自己猴群的猴王，每只猴子有一个初始的力量值。这些猴子会有M次会面。每次两只猴子x,y会面，若x,y属于同一个猴群输出-1，否则将x,y所在猴群的猴王的力量值减半，然后合并这两个猴群。新猴群中力量值最高的为猴王。输出新猴王的力量值。

分析：涉及集合的查询，合并，取最值。 利用并查集和左偏树即可解决。

看代码：

#include<cstdio>

#include<iostream>

#include<math.h>

using namespace std;

const int maxn=2e5;

int tot,v[maxn],l[maxn],r[maxn],d[maxn],fa[maxn];

int Findset(int x)

{

if(fa[x]==x) return fa[x];

return fa[x]=Findset(fa[x]);

}

void Init(int x)//初始化{

tot++;

v[tot]=x;//存对应的值

fa[tot]=tot;//并查集初始化

l[tot]=r[tot]=d[tot]=0;//左子树和右子树和距离为0

}

int Merge(int x,int y){//两棵树合并

if(!x) return y;//找到插入的地方了

if(!y) return x;

if(v[x]<v[y]) swap(x,y);//使得根节点最大

r[x]=Merge(r[x],y);//递归合并右子树和y

fa[r[x]]=x;//更新右子树的根

if(d[l[x]]<d[r[x]]) swap(l[x],r[x]);

d[x]=d[r[x]]+1;

return x;//新的根

}

int Pop(int x)

{

int L=l[x];//取出左子树

int R=r[x];//取出右子树

fa[L]=L;//根变为自己

fa[R]=R;

v[x]/=2;//取根节点 值除二

r[x]=l[x]=d[x]=0;

return Merge(L,R);//左右子树合并 返回值为左右子树合并后新的根

}

int Top(int x)

{

return v[x];

}

void solve(int x,int y)

{

int Left=Pop(x);//去掉x后 左右子树合并后新的根

int Right=Pop(y);//去掉y后 左右子树合并后新的根

Left=Merge(Left,x);//把更新后的值重新加入该树中

Right=Merge(Right,y);

Left=Merge(Left,Right);//两棵树合并

printf("%d\n",Top(Left));//两棵树合并之后的根}

int main()

{

int n,m,i,x,y;

while(scanf("%d",&n)!=EOF)

{

tot=0;

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

{

scanf("%d",&x);

Init(x);

}

scanf("%d",&m);

for(i=1;i<=m;i++)

{

scanf("%d%d",&x,&y);

int fx=Findset(x);

int fy=Findset(y);

if(fx==fy) printf("-1\n");

else solve(fx,fy);

}

}

return 0;

}