左偏树：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108 | #include<bits/stdc++.h>  #define LiangJiaJun main  using namespace std;    multiset<int> st;  void Erase(int x){      st.erase(st.find(x));  }  void Insert(int x){      st.insert(x);  }  struct Sheap{      int ls[300004],rs[300004],fa[300004];      int tag[300004],v[300004],q[300004];      int tot;      int reset(){          memset(ls,0,sizeof(ls));          memset(rs,0,sizeof(rs));          memset(fa,0,sizeof(fa));          memset(tag,0,sizeof(tag));          memset(v,0,sizeof(v));          memset(q,0,sizeof(q));          tot=0;      }      int Find(int x){          while(fa[x])x=fa[x];          return x;      }      void pushdown(int k){          if(!tag[k])return;          int t=tag[k],l=ls[k],r=rs[k];          tag[k]=0;          if(l)tag[l]+=t,v[l]+=t;          if(r)tag[r]+=t,v[r]+=t;      }      void solvetag(int x){          while(x)q[++tot]=x,x=fa[x];          while(tot)pushdown(q[tot--]);      }      int top(int x){          return v[Find(x)];      }      int Merge(int x,int y){          if(!x||!y)return x+y;          if(v[x]<v[y])swap(x,y);          pushdown(x);          rs[x]=Merge(rs[x],y);          fa[rs[x]]=x;          swap(ls[x],rs[x]);          return x;      }      int del(int x){          solvetag(x);          int t=Merge(ls[x],rs[x]),f=fa[x];          ls[x]=rs[x]=fa[x]=0;          if(x==ls[f])ls[f]=t;          else rs[f]=t;          fa[t]=f;          return Find(t);      }      void add(int x,int val){          solvetag(x);          Erase(v[Find(x)]);          v[x]+=val;          Insert(v[Merge(x,del(x))]);      }      void heap\_add(int x,int val){          int p=Find(x);          tag[p]+=val;v[p]+=val;          Erase(v[p]-val);Insert(v[p]);      }      void addedge(int x,int y){          int t1=Find(x),t2=Find(y);          if(t1!=t2){              if(Merge(t1,t2)==t1)st.erase(st.find(v[t2]));              else st.erase(st.find(v[t1]));          }      }  }q;  int n,m;  int w33ha(int CASE){      scanf("%d",&n);      st.clear();q.reset();      for(int i=1;i<=n;i++){          scanf("%d",&q.v[i]);          st.insert(q.v[i]);      }      scanf("%d",&m);      printf("Case #%d:\n",CASE);      for(int i=1;i<=m;i++){          int x,y;          scanf("%d%d",&x,&y);          if(q.Find(x)==q.Find(y)){puts("-1");continue;}          else{              int nowx=q.Find(x),nowy=q.Find(y);              q.add(nowx,-q.v[nowx]+(q.v[nowx]>>1));              q.add(nowy,-q.v[nowy]+(q.v[nowy]>>1));              q.addedge(x,y);              printf("%d\n",q.top(x));          }      }      return 0;  }  int LiangJiaJun(){      int T;scanf("%d",&T);      for(int i=1;i<=T;i++)w33ha(i);      return 0;  } |