**树上莫队**

You are given a tree with N nodes. The tree nodes are numbered from 1 to N. Each node has an integer weight.

We will ask you to perform the following operation:

u v : ask for how many different integers that represent the weight of nodes there are on the path from u to v.

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int const SIZE=40100;  int const BLOCK\_SIZE=300;  //利用hash记录LCA  struct Hash{  typedef struct \_\_t{int a;int b;\_\_t(int aa=0,int bb=0):a(aa),b(bb){}}key\_t;  typedef int value\_t;  enum{MOD=0x1fffff};  key\_t keys[MOD+1];  value\_t values[MOD+1];  int head[MOD+1];  int next[MOD+1];  int toUsed;  Hash():toUsed(0){fill(head,head+MOD+1,-1);}  void clear(){fill(head,head+MOD+1,-1);toUsed=0;}  int getKey(key\_t const&key)const {  int ret=17;  ret=ret\*37+key.a;  ret=ret\*37+key.b;  return ret;  }  void insert(key\_t const&key,value\_t const&value){  int k = getKey(key) & MOD;  keys[toUsed] = key;  values[toUsed] = value;  next[toUsed] = head[k];  head[k] = toUsed++;  }  value\_t find(key\_t const&key)const{  int k = getKey(key) & MOD;  for(int i=head[k];i!=-1;i=next[i]){  if ( keys[i].a == key.a && keys[i].b == key.b ) return values[i];  }  return 0;  }  void disp(FILE\*fp)const{  for(int i=1;i<toUsed;++i){  fprintf(fp,"(%d %d): %d\n",keys[i].a,keys[i].b,values[i]);  }  }  }Lca;  struct dege\_t{  int to;  int next;  }Edge[SIZE<<1];  int ECnt=1;  int Vertex[SIZE]={0};  inline void makeEdge(int a,int b)  {  Edge[ECnt].to=b;  Edge[ECnt].next=Vertex[a];  Vertex[a]=ECnt++;  Edge[ECnt].to=a;  Edge[ECnt].next=Vertex[b];  Vertex[b]=ECnt++;  }  //生成DFS序  int InIdx[SIZE],OutIdx[SIZE];  int NewIdx[SIZE<<1];  int NCnt = 1;  void dfs(int node,int parent){  NewIdx[NCnt] = node;  InIdx[node] = NCnt++;  for(int next=Vertex[node];next;next=Edge[next].next){  int to = Edge[next].to;  if ( to != parent ) dfs(to,node);  }  NewIdx[NCnt] = node;  OutIdx[node] = NCnt++;  }  //Tarjan算法中用到的并查集  int Father[SIZE];  int find(int x){return x==Father[x]?x:Father[x]=find(Father[x]);}  bool Flag[SIZE] = {false};  vector<vector<int> > Questions(SIZE,vector<int>());  //Tarjan算法一次性求出所有的LCA  void Tarjan(int u,int parent){  Father[u] = u;  Flag[u] = true;  for(int next=Vertex[u];next;next=Edge[next].next){  int to = Edge[next].to;  if ( to == parent ) continue;  Tarjan(to,u);  Father[to] = u;  }  vector<int>&vec=Questions[u];  for(vector<int>::iterator it=vec.begin();it!=vec.end();++it){  int v = \*it;  if ( Flag[v] ){  int r = find(v);  Lca.insert(Hash::key\_t(u,v),r);  Lca.insert(Hash::key\_t(v,u),r);  }  }  }  struct \_t{  int s,e;  int idx;  int lca;  };  bool operator < (\_t const&lhs,\_t const&rhs){  int ln = lhs.s / BLOCK\_SIZE;  int rn = rhs.s / BLOCK\_SIZE;  return ln < rn || ( ln == rn && lhs.e < rhs.e );  }  int N,M;  int A[SIZE];  \_t B[100000];  //将原树上的路径问题转化为DFS序中的区间问题  inline void mkQuestion(int a,int b,int idx){  int lca = Lca.find(Hash::key\_t(a,b));  if ( lca == a || lca == b ){  int t = lca == a ? b : a;  B[idx].s = OutIdx[t];  B[idx].e = OutIdx[lca];  B[idx].lca = 0;  }else{  B[idx].lca = lca;  if ( OutIdx[a] < InIdx[b] ) B[idx].s = OutIdx[a], B[idx].e = InIdx[b];  else B[idx].s = OutIdx[b], B[idx].e = InIdx[a];  }  }  int MoAns;  int Ans[100000],Cnt[SIZE];  inline void insert(int n){  if ( 1 == ++Cnt[n] ) ++MoAns;  }  inline void remove(int n){  if ( 0 == --Cnt[n] ) --MoAns;  }  void MoOp(int idx){  int k = NewIdx[idx];  if ( Flag[k] ) remove(A[k]);  else insert(A[k]);  Flag[k] ^= 1;  }  void Mo(){  sort(B,B+M);  fill(Flag,Flag+N+1,false);  int curLeft = 1;  int curRight = 0;  MoAns = 0;  for(int i=0;i<M;++i){  while( curRight < B[i].e ) MoOp(++curRight);  while( curLeft > B[i].s ) MoOp(--curLeft);  while( curRight > B[i].e ) MoOp(curRight--);  while( curLeft < B[i].s ) MoOp(curLeft++);  if ( B[i].lca ){  Ans[B[i].idx] = MoAns + ( 0 == Cnt[A[B[i].lca]] ? 1 : 0 );  }else{  Ans[B[i].idx] = MoAns;  }  }  }  void init(int n){  ECnt = NCnt = 1;  fill(Vertex,Vertex+n+1,0);  fill(Flag,Flag+n+1,false);  }  int getUnsigned(){  char ch = getchar();  while( ch > '9' || ch < '0' ) ch = getchar();  int ret = 0;  do ret = ret \* 10 + (int)(ch-'0');while( '0' <= (ch=getchar()) && ch <= '9' );  return ret;  }  int W[SIZE];  bool read(){  if ( EOF == scanf("%d",&N) ) return false;  M = getUnsigned();  init(N);  //权值输入并离散化  for(int i=1;i<=N;++i) W[i] = A[i] = getUnsigned();  sort(W+1,W+N+1);  int\* pn = unique(W+1,W+N+1);  for(int i=1;i<=N;++i) A[i] = lower\_bound(W+1,pn,A[i]) - W;  int a,b;  for(int i=1;i<N;++i){  a = getUnsigned();  b = getUnsigned();  makeEdge(a,b);  }  dfs(1,0);  for(int i=0;i<M;++i){  B[i].s = getUnsigned();  B[i].e = getUnsigned();  B[i].idx = i;  Questions[B[i].s].push\_back(B[i].e);  Questions[B[i].e].push\_back(B[i].s);  }  Tarjan(1,0);  for(int i=0;i<M;++i) mkQuestion(B[i].s,B[i].e,i);  return true;  }  int main(){  //freopen("1.txt","r",stdin);  while ( read() ){  Mo();  for(int i=0;i<M;++i)printf("%d\n",Ans[i]);  }  return 0;  } |

**带修改的莫队算法**

有n个数编号从0→n-1,两种操作：   
Q L R:询问编号为L→R-1的数中共有多少种不同的数   
M X Y：将编号为X的数改为Y   
共有m个操作

|  |
| --- |
| #include<iostream>  #include<cstdio>  #include<cstring>  #include<algorithm>  #include<cmath>  #define fo(i,a,b) for(i=a;i<=b;i++)  #define fod(i,a,b) for(i=a;i>=b;i--)  using namespace std;  const int maxn=50007;  struct node{  int l,r,x,p;  }a[maxn];  struct nod{  int x,y,o;  }d[maxn];  int i,j,k,l,t,n,m,ans,now,kuai[maxn],da,last[maxn],tot,num,r;  bool bz[maxn];  int shu[1000007],b[maxn],ans1[maxn];  char s[10];  bool cmp(node x,node y){  return kuai[x.l]<kuai[y.l]||kuai[x.l]==kuai[y.l]&&x.r<y.r||kuai[x.l]==kuai[y.l]&&x.r==y.r&&x.x<y.x;  }  void update(int x){  if(bz[x]){  shu[b[x]]--;  if(!shu[b[x]])ans--;  }  else{  shu[b[x]]++;  if(shu[b[x]]==1)ans++;  }  bz[x]^=1;  }  void change(int x,int y){  if(bz[x]){  update(x);b[x]=y;update(x);  }  else b[x]=y;  }  int main(){  freopen("fan.in","r",stdin);  freopen("fan.out","w",stdout);  scanf("%d%d",&n,&m);  da=500;  fo(i,1,n)scanf("%d",&b[i]),kuai[i]=i/da,last[i]=b[i];  fo(i,1,m) {  scanf("%s%d%d",s,&k,&l);  if (s[0]=='Q')a[++tot].l=k,a[tot].r=l,a[tot].x=num,a[tot].p=tot;  else d[++num].x=k,d[num].o=last[k],d[num].y=l,last[k]=l;  }  sort(a+1,a+1+tot,cmp);  l=1;  fo(i,1,tot){  if (now<a[i].x)fo(j,now+1,a[i].x)change(d[j].x,d[j].y);  else fod(j,now,a[i].x+1)change(d[j].x,d[j].o);  if (l<a[i].l)fo(j,l,a[i].l-1)update(j);  else fo(j,a[i].l,l-1)update(j);  if (r<a[i].r)fo(j,r+1,a[i].r)update(j);  else fo(j,a[i].r+1,r)update(j);  ans1[a[i].p]=ans;l=a[i].l;r=a[i].r;now=a[i].x;  }  fo(i,1,tot)printf("%d\n",ans1[i]);  } |