Backlight's Code Template

Backl1ght

March 18, 2020

Central South University
blog: https://www.cnblogs.com/zengzk/
E-mail: 1229309527@qq.com

Content

1 基本

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef double db;
typedef pair<int,int> PI;
typedef vector<int> VI;

#define rep(i,_,__) for (int i=_; i<=__;i++)
#define per(i, _, __) for (int i = _; i >= __;i--)
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define x1 _x
#define x2 __x
#define y1 _y
#define y2 __y
#define sz(x) ((int)(x).size())
#define all(x) (x).begin(),(x).end()
#define rall(x) (x).rbegin(),(x).rend()
#define endl '\n'

const double pi=acos(-1.0);

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
int rnd(int l,int r){return l+rng()%(r-l+1);}

namespace IO{
   bool REOF = 1; //为0表示文件结尾
   inline char nc() {
      static char buf[100000], *p1 = buf, *p2 = buf;
      return p1 == p2 && REOF && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? (REOF = 0, EOF) :
          *p1++;
   }

   template<class T>
   inline bool read(T &x) {
      char c = nc();bool f = 0; x = 0;
      while (c<'0' || c>'9')c == '-' && (f = 1), c = nc();
      while (c >= '0'&&c <= '9')x = (x << 3) + (x << 1) + (c ^ 48), c = nc();
      if(f)x=-x;
      return REOF;
   }

   template<typename T, typename... T2>
   inline bool read(T &x, T2 &... rest) {
      read(x);
      return read(rest...);
   }

   inline bool need(char &c) { return ((c >= 'a') && (c <= 'z')) || ((c >= '0') && (c <= '9')) || ((c >= 'A') &&
        (c <= 'Z')); }
   // inline bool need(char &c) { return ((c >= 'a') && (c <= 'z')) || ((c >= '0') && (c <= '9')) || ((c >= 'A')
        && (c <= 'Z')) || c==' '; }

   inline bool read_str(char *a) {
      while ((*a = nc()) && need(*a) && REOF)++a; *a = '\0';
      return REOF;
   }

   inline bool read_dbl(double &x){
      bool f = 0; char ch = nc(); x = 0;
```

```
        while(ch<'0'||ch>'9') {f|=(ch=='-');ch=nc();}
        while(ch>='0'&&ch<='9'){x=x*10.0+(ch^48);ch=nc();}
        if(ch == '.') {
            double tmp = 1; ch = nc();
            while(ch>='0'&&ch<='9'){tmp=tmp/10.0;x=x+tmp*(ch^48);ch=nc();}
        }
        if(f)x=-x;
        return REOF;
    }

    template<class TH> void _dbg(const char *sdbg, TH h){ cerr<<sdbg<<'='<<h<<endl; }

    template<class TH, class... TA> void _dbg(const char *sdbg, TH h, TA... a) {
        while(*sdbg!=',')cerr<<*sdbg++;
        cerr<<'='<<h<<','<<' '; _dbg(sdbg+1, a...);
    }

    template<class T> ostream &operator<<(ostream& os, vector<T> V) {
        os << "["; for (auto vv : V) os << vv << ","; return os << "]";
    }

    template<class T> ostream &operator<<(ostream& os, set<T> V) {
        os << "["; for (auto vv : V) os << vv << ","; return os << "]";
    }

    template<class T> ostream &operator<<(ostream& os, map<T,T> V) {
        os << "["; for (auto vv : V) os << vv << ","; return os << "]";
    }

    template<class L, class R> ostream &operator<<(ostream &os, pair<L,R> P) {
        return os << "(" << P.x << "," << P.y << ")";
    }

    #define debug(...) _dbg(#__VA_ARGS__, __VA_ARGS__)
}

using namespace IO;
const int maxn = 5e5 + 5;
const int maxv = 1e7 + 5;
const int mod = 998244353; // 998244353 1e9+7
const int INF = 0x3f3f3f3f; // 1e9+7 0x3f3f3f3f
const ll LINF = 1e18+9; // 1e18+9 0x3f3f3f3f3f3f3f3f
const double eps = 1e-12;

int dx[4] = { 0, 1, 0, -1 };
// int dx[8] = { 1, 0, -1, 1, -1, 1, 0, -1 };
int dy[4] = { 1, 0, -1, 0 };
// int dy[8] = { 1, 1, 1, 0, 0, -1, -1, -1 };

// ll qp(ll a, ll b) {
// ll res = 1;
// a %= mod;
// assert(b >= 0);
// while(b){
// if(b&1)
// res = res * a % mod;
// a = a * a % mod;
// b >>= 1;
// }
// return res;
// }

// #define ls (x<<1)
// #define rs (x<<1|1)
// #define mid ((l+r)>>1)
// #define lson ls,l,mid
// #define rson rs,mid+1,r

// int tot = 1, head[maxv];
// struct Edge{
//  int v,nxt;
// Edge(){}
// Edge(int _v,int _nxt):v(_v),nxt(_nxt){}
// } e[maxv << 1];
// void init(){
// tot = 1;
// memset(head, 0, sizeof(head));
// }
// void addedge(int u,int v){
// e[tot] = Edge(v, head[u]);
// head[u] = tot++;
// e[tot] = Edge(u, head[v]);3.
// head[v] = tot++;
// }
// void addarc(int u,int v){
// e[tot] = Edge(v, head[u]);
// head[u] = tot++;
// }

/**
* ********** Backlight **********
* 仔细读题
* 注意边界条件
* 记得注释输入流重定向
* 没有思路就试试逆向思维
* 加油，奥利给
*/

void solve(int Case){

}

int main()
{
   // freopen("in.txt", "r", stdin);
   // ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
   // int _T; read(_T); rep(_, 1, _T) solve(_);
   // while(read(n)) solve();
   // solve(1);
   return 0;
}
```

```
//手动扩栈 O2优化
#pragma comment(linker, "/STACK:102400000,102400000")
#pragma GCC optimize(2)
```

2 字符串

2.1 字符串 HASH

```
namespace Hash{
   typedef unsigned long long ull;
   const int BASE=29;
   ull p[maxn];
   void init(){
      p[0] = 1;
      for(int i=1; i<maxn; i++) p[i] = p[i-1] * BASE;
   }
   void get_hash(char* a, ull* h, int len){
        h[0]=len;
        for(long i=1; i<=len; i++)h[i] = h[i-1] * BASE + a[i] - 'a' + 1;
    }
    ull get_code(ull* h, int l, int r){
        return h[r]-h[l-1]*p[r-l+1];
    }
}
```

2.2 manacher

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 11000009;
char s[maxn];

int len[maxn<<1],N;
char ch[maxn<<1];
void init(char *s){
    int n = strlen(s+1);//s:range[1,n]
    ch[n*2 +1] = '#';
    ch[0] = '@';
    ch[n*2 +2] = '\0';
    for (int i=n;i>=1;i--)ch[i*2] = s[i], ch[i*2 -1] = '#';
    N = 2* n +1;
}
void manacher(){
    int ma=0, k=1;
    len[1]=1;
    for (int i=2;i<=N;i++){
        int p = k+len[k]-1;
        if (i<=p)len[i]=min(len[2*k-i],p-i+1);
        else len[i]=1;
        while (ch[i+len[i]]==ch[i-len[i]])len[i]++;
        if (i+len[i]>k+len[k])k=i;
        ma=max(ma,len[i]);
    }
    printf("%d\n",ma-1);
}
int main(){
    scanf("%s",s+1);
    init(s);
    manacher();
    return 0;
}
```

2.3 trie

```
namespace Trie{
    // string index range: [1, len]
    // read(s+1), and use func(s)
    int sz;
    int ch[maxn][26];
    int val[maxn];// 记录某些特性，如以当前字符串为结尾的串的个数
    void init() {
        sz=1;
        memset(ch[0], 0, sizeof(ch[0]));
        memset(val, 0, sizeof(val));
    }
    inline int idx(char c){return c-'a';}
    void add(char* s) {
        int u=0, len=strlen(s+1);
        for(int i=1; i<=len; i++){
            int c = idx(s[i]);
            if(!ch[u][c]) {
                memset(ch[sz], 0, sizeof(ch[sz]));
                val[sz] = 0;
                ch[u][c] = sz++;
            }
            u = ch[u][c];
            val[u]++;
        }
    }
    int query(char* s) {
        int u=0, len=strlen(s+1);
        for(int i=1; i<=len; i++){
            int c = idx(s[i]);
            if(!ch[u][c]) return 0;
            u = ch[u][c];
        }
        return val[u];
    }
}
```

2.4 可持久化 trie

$$\begin{aligned} &given\ an\ array\ a\ and\ some\ (l,r,k), calc \\ &ans = max\left\{k \oplus_{i=p}^{n} a_i\right\}\ \text{for}\ l \le p \le r \end{aligned} \tag{1}$$

```
//转化为[l-1,r-1]中异或(sn^k)的最大值
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 600010;
int n, q, a[maxn], s[maxn], l, r, x;
char op;
struct Trie {
    int cnt, rt[maxn], ch[maxn * 33][2], val[maxn * 33];
    void insert(int o, int lst, int v) {
        for (int i = 28; i >= 0; i--) {
            val[o] = val[lst] + 1; //在原版本的基础上更新
            if ((v & (1 << i)) == 0) {
                if (!ch[o][0]) ch[o][0] = ++cnt;
                ch[o][1] = ch[lst][1];
                o = ch[o][0];
                lst = ch[lst][0];
            } else {
                if (!ch[o][1]) ch[o][1] = ++cnt;
                ch[o][0] = ch[lst][0];
                o = ch[o][1];
                lst = ch[lst][1];
            }
        }
        val[o] = val[lst] + 1;
        // printf("%d\n",o);
    }
    int query(int o1, int o2, int v) {
        int ret = 0;
        for (int i = 28; i >= 0; i--) {
            // printf("%d %d %d\n",o1,o2,val[o1]-val[o2]);
            int t = ((v & (1 << i)) ? 1 : 0);
            if (val[ch[o1][!t]] - val[ch[o2][!t]])
            ret += (1 << i), o1 = ch[o1][!t], o2 = ch[o2][!t]; //尽量向不同的地方跳
            else
            o1 = ch[o1][t], o2 = ch[o2][t];
        }
        return ret;
    }
} st;
int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i++) scanf("%d", a + i), s[i] = s[i - 1] ^ a[i];
    for (int i = 1; i <= n; i++)
    st.rt[i] = ++st.cnt, st.insert(st.rt[i], st.rt[i - 1], s[i]);
    while (q--) {
        scanf(" %c", &op);
        if (op == 'A') {
            n++;
            scanf("%d", a + n);
            s[n] = s[n - 1] ^ a[n];
            st.rt[n] = ++st.cnt;
            st.insert(st.rt[n], st.rt[n - 1], s[n]);
        }
        if (op == 'Q') {
            scanf("%d%d%d", &l, &r, &x);
            l--;
            r--;
            if (l == r && l == 0)
            printf("%d\n", s[n] ^ x); //记得处理 l=r=1 的情况
            else
            printf("%d\n", st.query(st.rt[r], st.rt[max(l - 1, 0)], x ^ s[n]));
        }
    }
    return 0;
}
```

2.5 KMP

```
// hdu 1711 返回第一次匹配的位置
#include <bits/stdc++.h>
using namespace std;

const int N = 1e4 + 5;
const int M = 1e6 + 5;
int s[N], t[M];
int n, m, nxt[N], fail[N];
void get_fail() {
    int k = 0;
    for (int i=1; i<=n; i++) {
        nxt[i] = k;
        fail[i] = (s[i]==s[k])?fail[k]:k;
        while(k && s[i] != s[k]) k = fail[k];
        if (s[i]==s[k]) k++;
    }
}

int match() {
    int pos = -1;
    int j =0;
    for (int i=0; i<m; i++) {
        while(j && t[i]!=s[j]) j=fail[j];
        if (t[i]==s[j]) {
            j++;
            if (j==n) {
                pos = i + 2 - n;
                break;
            }
        }
    }
    return pos;
}

int repete() {
    int len = m - nxt[m];
    return m%len?m:len;
}

void solve(int Case) {
    scanf("%d %d", &m, &n);
    for(int i=0; i<m; i++) scanf("%d", &t[i]);
    for(int i=0; i<n; i++) scanf("%d", &s[i]);
    get_fail();
    printf("%d\n", match());
}

int main()
{
    int T; scanf("%d", &T); for(int i=1;i<=T;i++) solve(i);
    return 0;
}
```

2.6 扩展 KMP

```
//extend[i]:文本串的后缀s(i,lens)和模式串t的最长公共前缀
void EKMP(char s[],char t[])//s为文本串，t为模式串
{
    int i,j,p,L;
    int lens=strlen(s);
    int lent=strlen(t);
    next[0]=lent;
    j=0;
    while(j+1<lent && t[j]==t[j+1])j++;
    next[1]=j;

    int a=1;
    for(i=2;i<lent;i++)
    {
        p=next[a]+a-1;
        L=next[i-a];
        if(i+L<p+1)next[i]=L;
        else
        {
            j=max(0,p-i+1);
            while(i+j<lent && t[i+j]==t[j])j++;
            next[i]=j;
            a=i;
        }
    }

    j=0;
    while(j<lens && j<lent && s[j]==t[j])j++;
    extend[0]=j;
    a=0;
    for(i=1;i<lens;i++)
    {
        p=extend[a]+a-1;
        L=next[i-a];
        if(L+i<p+1)extend[i]=L;
        else
        {
            j=max(0,p-i+1);
            while(i+j<lens && j<lent && s[i+j]==t[j])j++;
            extend[i]=j;
            a=i;
```

```
      }
   }
}
```

2.7 AC 自动机 1

```
//小于n且不包含任何模式串的数的个数
//fail树上跑数位DP
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=2e5+5;
const int mod=1e9+7;
int len,n;
char s[2019],t[2019];
struct AC_Automaton{
   //root=0,range[1,tot]
   const static int SIZE=10;
   int tr[maxn][SIZE],fail[maxn],tot;

   int val[maxn];
   ll dp[2019][2019];

   inline int newnode(){
      int p=++tot;
      memset(tr[p],0,sizeof(tr[p]));
      val[p]=0;
      return p;
   }
   inline void init(){
      tot=0;
      memset(tr[0],0,sizeof(tr[0]));
      val[0]=0;
   }
   inline void insert(char *s){
      int p=0;
      for(int i=0;s[i];++i){
         if(!tr[p][s[i]-'0'])tr[p][s[i]-'0']=newnode();
         p=tr[p][s[i]-'0'];
      }
      val[p]=1;
   }
   inline void getfail(){
      queue<int>q;
      for(int i=0;i<SIZE;i++)if(tr[0][i])fail[tr[0][i]]=0,q.push(tr[0][i]);
      while(!q.empty()){
         int p=q.front();q.pop();
         for(int i=0;i<SIZE;i++){
            if(tr[p][i]){
               fail[tr[p][i]]=tr[fail[p]][i],q.push(tr[p][i]);
            }
            else tr[p][i]=tr[fail[p]][i];
         }
      }
   }
   ll dfs(int p,int pos,int jud,int zero){
      if(pos==len)return !zero;
      if(!jud && dp[p][pos]!=-1)return dp[p][pos];
      int sz=jud?t[pos]-'0':9;
      ll res=0;
      for(int i=0;i<=sz;i++){
         int tmp=tr[p][i];
         if(val[tmp])continue;
         res+=dfs(zero?tr[0][i]:tmp,pos+1,jud&&(i==sz),zero&&(i==0));
```

```
            res%=mod;
        }
        if(!jud && !zero)dp[p][pos]=res;
        return res;
    }
    inline void solve(char *t){
        len=strlen(t);
        memset(dp,-1,sizeof(dp));
        printf("%lld\n",dfs(0,0,1,1));//减去0
    }
}A;

int main()
{
    A.init();
    scanf("%s",t);
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        scanf("%s",s);
        A.insert(s);
    }
    A.getfail();
    A.solve(t);
    return 0;
}
```

2.8 AC 自动机 2

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=2e4+10;
const int maxm=5e4+10;
const int N=1e5+10;
typedef map<int,int>::iterator MI;
typedef vector<int>::iterator VI;

vector<int>s[maxm],t[2*maxn];
struct AC_Automaton{
    int fail[N],tot;
    map<int,int>ch[N];

    vector<int>belong[N];
    int ans1[maxm],ans2[maxn],vis[maxm],flag[N];

    inline void init(){
        tot=0;
        ch[0].clear();
        belong[0].clear();
    }
    inline int newnode(){
        ++tot;
        ch[tot].clear();
        belong[tot].clear();
        return tot;
    }

    inline void insert(int id,int blg){
        int p=0,sz=(int)s[id].size();
        for(int i=0;i<sz;++i){
            int c=s[id][i];
            if(!ch[p][c])ch[p][c]=newnode();
            p=ch[p][c];
        }
        belong[p].push_back(blg);
    }
    inline void getfail(){
        queue<int>q;
        fail[0]=0;
        for(MI it=ch[0].begin();it!=ch[0].end();++it)
        q.push(it->second),fail[it->second]=0;
        while(!q.empty()){
            int p=q.front();q.pop();
            for(MI it=ch[p].begin();it!=ch[p].end();++it){
                int v=it->second,c=it->first,u=fail[p];
                while(u && !ch[u][c])u=fail[u];
                fail[v]=ch[u][c];
                q.push(v);
            }
        }
    }
    inline void solve(int id,int blg){
        int p=0,sz=(int)t[id].size();
        for(int i=0;i<sz;++i){
            int c=t[id][i];
            while(p && !ch[p][c])p=fail[p];
            p=ch[p][c];
            int tmp=p;
            while(tmp){
                if(flag[tmp]==blg){tmp=fail[tmp];continue;}
                flag[tmp]=blg;
                for(VI ii=belong[tmp].begin();ii!=belong[tmp].end();++ii)
                if(vis[*ii]!=blg){
                    ++ans2[blg];++ans1[*ii];vis[*ii]=blg;
                }
                tmp=fail[tmp];
            }
        }
    }
    inline void print(int n,int m){
        for(int i=1;i<=m;++i)printf("%d\n",ans1[i]);
        for(int i=1;i<n;++i)printf("%d ",ans2[i]);
        printf("%d\n",ans2[n]);
    }
}A;

int main()
{
    int n,m,len,x;
    A.init();
    scanf("%d %d",&n,&m);
    for(int i=1;i<=2*n;++i){
        scanf("%d",&len);
        for(int j=0;j<len;++j)scanf("%d",&x),t[i].push_back(x);
    }
    for(int i=1;i<=m;++i){
        scanf("%d",&len);
        for(int j=0;j<len;++j)scanf("%d",&x),s[i].push_back(x);
        A.insert(i,i);
    }
    A.getfail();
    for(int i=1;i<=n;++i){
        A.solve(2*i-1,i);
        A.solve(2*i,i);
    }
    A.print(n,m);
    return 0;
}
```

2.9 后缀数组倍增

```
/*
sa[i]:从sa[i]开始的后缀排名为i
rank[i]:从i开始的后缀排名为rank[i]
height[i]:排名为i-1和i的后缀的最长公共前缀

所有串中的最大重复次数
for(L=1;L <= n;L++){
   for (int i = 1; i + L <= n; i += L){
      int R = lcp(i, i + L);
      ans = max(ans, R / L + 1);
      if (i >= L - R % L)ans = max(lcp(i - L + R%L, i + R%L) / L + 1, ans);
   }
}
*/
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=2e6+5;

char s[maxn];
int height[maxn],c[maxn],x[maxn],y[maxn],sa[maxn],rk[maxn];
void SA(int n){
   n++;
   int i,j,k,m=1000;//m为字符集大小 ,max(s[i])<m
   for(i=0;i<m;i++) c[i]=0;
   for(i=0;i<n;i++)c[x[i]=s[i]]++;
   for(i=1;i<m;i++) c[i]+=c[i-1];
   for(i=n-1;i>=0;i--) sa[--c[x[i]]]=i;
   for(j=1;j<=n;j<<=1){
      k=0;
      for(i=n-j;i<n;i++) y[k++]=i;
      for(i=0;i<n;i++) if(sa[i]>=j) y[k++]=sa[i]-j;
      for(i=0;i<m;i++) c[i]=0;
      for(i=0;i<n;i++) c[x[y[i]]]++;
      for(i=1;i<m;i++) c[i]+=c[i-1];
      for(i=n-1;i>=0;i--) sa[--c[x[y[i]]]]=y[i];
      swap(x,y);
      m=0;
      x[sa[0]]=m++;
      for(i=1;i<n;i++){
         if(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+j]==y[sa[i-1]+j]) x[sa[i]]=m-1;
         else x[sa[i]]=m++;
      }
      if(m>=n) break;
   }
   k=0;
   for(i=0;i<n;i++) rk[sa[i]]=i;
   for(i=0;i<n-1;i++){
      if(k) k--;
      j=sa[rk[i]-1];
      while(s[i+k]==s[j+k]) k++;
      height[rk[i]]=k;
   }
}

int main()
{
   scanf("%s",s);
   int len=strlen(s);
   SA(len);
   ll ans=0;
   for(int i=1;i<=len;i++){
        ans+=len-sa[i]-height[i];
    }
    printf("%lld\n",ans);
    return 0;
}
```

2.10 后缀数组 SAIS

```
//O(n)
#include<bits/stdc++.h>
using namespace std;
const int N=1e6+5;
char S[N];
int n,m;
int s[N<<1],t[N<<1],height[N],sa[N],rk[N],p[N],c[N],w[N];
inline int trans(int n,const char* S){
    int m=*max_element(S+1,S+1+n);
    for(int i=1;i<=n;++i) rk[S[i]]=1;
    for(int i=1;i<=m;++i) rk[i]+=rk[i-1];
    for(int i=1;i<=n;++i) s[i]=rk[S[i]];
    return rk[m];
}
#define ps(x) sa[w[s[x]]--]=x
#define pl(x) sa[w[s[x]]++]=x
inline void radix(int* v,int* s,int* t,int n,int m,int n1){
    memset(sa,0,n+1<<2); memset(c,0,m+1<<2);
    for(int i=1;i<=n;++i) ++c[s[i]];
    for(int i=1;i<=m;++i) w[i]=c[i]+=c[i-1];
    for(int i=n1;i;--i) ps(v[i]);
    for(int i=1;i<=m;++i) w[i]=c[i-1]+1;
    for(int i=1;i<=n;++i) if(sa[i]>1 && t[sa[i]-1]) pl(sa[i]-1);
    for(int i=1;i<=m;++i) w[i]=c[i];
    for(int i=n;i;--i) if(sa[i]>1 && !t[sa[i]-1]) ps(sa[i]-1);
}
inline void SAIS(int n,int m,int* s,int* t,int* p){
    int n1=0,ch=rk[1]=0,*s1=s+n; t[n]=0;
    for(int i=n-1;i;--i) t[i]=s[i]==s[i+1]?t[i+1]:s[i]>s[i+1];
    for(int i=2;i<=n;++i) rk[i]=t[i-1]&&!t[i]?(p[++n1]=i,n1):0;
    radix(p,s,t,n,m,n1);
    for(int i=1,x,y;i<=n;++i) if(x=rk[sa[i]]){
        if(ch<=1 || p[x+1]-p[x]!=p[y+1]-p[y]) ++ch;
        else for(int j=p[x],k=p[y];j<=p[x+1];++j,++k)
        if((s[j]<<1|t[j])^(s[k]<<1|t[k])){ ++ch; break; }
        s1[y=x]=ch;
    }
    if(ch<n1) SAIS(n1,ch,s1,t+n,p+n1);
    else for(int i=1;i<=n1;++i) sa[s1[i]]=i;
    for(int i=1;i<=n1;++i) s1[i]=p[sa[i]];
    radix(s1,s,t,n,m,n1);
}
inline void SA(int n,const char* S){
    int m=trans(++n,S); SAIS(n,m,s,t,p);
    for(int i=1;i<n;++i) rk[sa[i]=sa[i+1]]=i;
    for(int i=1,j,k=0;i<n;++i) if(rk[i]>1){
        for(j=sa[rk[i]-1];S[i+k]==S[j+k];++k);
        if(height[rk[i]]=k) --k;
    }
}
int main(){
    scanf("%s",S+1); n=strlen(S+1); SA(n,S);
    for(int i=1;i<=n;i++)printf("%d ",sa[i]);
}
```

2.11 后缀自动机 1

```
//广义后缀自动机：insert后重新将last赋1
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=1e6+5;

char s[maxn];
struct Suffix_Automaton
{
    //初始状态为0,range[0...tot-1]
    struct state{
        int len,link;
        map<char,int>next;
    }st[maxn<<1];
    int last,tot;

    void init(){
        st[0].len=0;st[0].link=-1;
        tot++;
        last=0;
    }

    void extend(char c){
        int cur=tot++;
        st[cur].len=st[last].len+1;
        int p=last;
        while(p!=-1 && !st[p].next.count(c)){
            st[p].next[c]=cur;
            p=st[p].link;
        }
        if(p==-1)st[cur].link=0;
        else{
            int q=st[p].next[c];
            if(st[p].len+1==st[q].len)st[cur].link=q;
            else{
                int clone=tot++;
                st[clone].len=st[p].len+1;
                st[clone].next=st[q].next;
                st[clone].link=st[q].link;
                while(p!=-1 && st[p].next[c]==q){
                    st[p].next[c]=clone;
                    p=st[p].link;
                }
                st[q].link=st[cur].link=clone;
            }
        }
        last=cur;
    }

    ll count(){
        ll res=0;
        for(int i=0;i<tot;i++)res+=st[i].len-st[st[i].link].len;
        return res;
    }
}sam;

int main()
{
    scanf("%s",s);
    sam.init();
    for(int i=0;s[i]!=0;i++)sam.extend(s[i]);
    printf("%lld\n",sam.count());
    return 0;
}
```

2.12 后缀自动机 2

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=2e5+5;
const ll INF=0x3f3f3f3f3f3f3f3fLL;

int n;
char s[maxn];
struct SAM{
    //range[1-tot],1为初始状态
    int ch[maxn<<1][26],fa[maxn<<1],len[maxn<<1];
    int last,tot;

    ll P,Q,dp[maxn];

    inline void init(){
        last=tot=1;
        len[1]=fa[0]=0;
        memset(ch[1],0,sizeof(ch[1]));
    }

    inline int newnode(){
        tot++;
        len[tot]=fa[tot]=0;
        memset(ch[tot],0,sizeof(ch[tot]));
        return tot;
    }

    inline void extend(int c){
        int p=last;
        int cur=newnode();
        len[cur]=len[last]+1;
        last=cur;
        while(p && !ch[p][c]){
            ch[p][c]=cur;
            p=fa[p];
        }
        if(!p)fa[cur]=1;
        else{
            int q=ch[p][c];
            if(len[p]+1==len[q])fa[cur]=q;
            else{
                int clone=newnode();
                len[clone]=len[p]+1;
                memcpy(ch[clone],ch[q],sizeof(ch[q]));
                fa[clone]=fa[q];
                fa[q]=fa[cur]=clone;
                while(ch[p][c]==q){
                    ch[p][c]=clone;
                    p=fa[p];
                }
            }
        }
    }

    inline void solve(int n){
        int j=1;
        memset(dp,0x3f,sizeof(dp));dp[1]=P;
        extend(s[1]-'a');
        int now=1;now=ch[now][s[1]-'a'];
        for(int i=2;i<=n;i++){
            dp[i]=dp[i-1]+P;
            while(true){
                while(now!=1 && len[fa[now]]>=i-j-1)now=fa[now];
                if(ch[now][s[i]-'a']){
                    now=ch[now][s[i]-'a'];
                    break;
                }
                else extend(s[++j]-'a');
            }
            dp[i]=min(dp[i],dp[j]+Q);
        }
        printf("%lld\n",dp[n]);
    }
}S;

int main()
{
    while(~scanf("%s",s+1)){
    S.init();
    scanf("%lld %lld",&S.P,&S.Q);
    //for(int i=1;i<=n;i++)S.extend(s[i]-'a');
    S.solve(strlen(s+1));
}
return 0;
}
```

2.13 SAM 线段树合并求 endpos 集

```
//给定串第k次出现的位置
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=1e5+5;
int n,q;
char s[maxn];

namespace SegTree{
    int sum[maxn<<6],L[maxn<<6],R[maxn<<6];
    int tot1;
    int update(int rt,int l,int r,int pos,int val){
        int nrt=++tot1;
        L[nrt]=L[rt]; R[nrt]=R[rt]; sum[nrt]=sum[rt]+val;
        if(l!=r){
            int mid=(l+r)>>1;
            if(pos<=mid)L[nrt]=update(L[rt],l,mid,pos,val);
            else R[nrt]=update(R[rt],mid+1,r,pos,val);
        }
        return nrt;
    }
    int merge(int rt1,int rt2){
        if(!rt1 || !rt2)return rt1|rt2;
        int nrt=++tot1;
        L[nrt]=L[rt1]; R[nrt]=R[rt1]; sum[nrt]=sum[rt1]+sum[rt2];
        L[nrt]=merge(L[rt1],L[rt2]);
        R[nrt]=merge(R[rt1],R[rt2]);
        return nrt;
    }

    int query(int rt,int l,int r,int k){
        if(l==r)return l;
        int mid=(l+r)>>1;
        if(k<=sum[L[rt]])return query(L[rt],l,mid,k);
        return query(R[rt],mid+1,r,k-sum[L[rt]]);
    }
}using namespace SegTree;

namespace Suffix_Automaton{
    int ch[maxn<<1][26],fa[maxn<<1],len[maxn<<1];
    int last,tot;

    int rt[maxn],T[maxn<<1];
    int Fa[maxn<<1][20];

    inline void init(){
        last=tot=1;
        len[1]=fa[0]=0;
        memset(ch[1],0,sizeof(ch[1]));

        T[1]=0;
    }

    inline int newnode(){
        ++tot;
        len[tot]=fa[tot]=0;
        memset(ch[tot],0,sizeof(ch[tot]));

        T[tot]=0;
        return tot;
    }

    inline void extend(int c,int right){
        int p=last,cur=newnode();
        len[cur]=len[last]+1;
        last=cur;

        rt[right]=cur;
        T[cur]=update(T[cur],1,n,right,1);

        while(p && !ch[p][c]){
            ch[p][c]=cur;
            p=fa[p];
        }
        if(!p)fa[cur]=1;
        else{
            int q=ch[p][c];
            if(len[p]+1==len[q])fa[cur]=q;
            else{
                int clone=newnode();
                len[clone]=len[p]+1;
                memcpy(ch[clone],ch[q],sizeof(ch[q]));
                fa[clone]=fa[q];
                fa[q]=fa[cur]=clone;
                while(ch[p][c]==q){
                    ch[p][c]=clone;
                    p=fa[p];
                }
            }
        }
    }

    int c[maxn<<1],A[maxn<<1];
    inline void init(char *a,int l){
        init();
        for(int i=1;i<=l;i++)extend(a[i]-'a',i);

        for(int i=0;i<=tot;i++)c[i]=0;
        for(int i=1;i<=tot;i++)++c[len[i]];
        for(int i=1;i<=tot;i++)c[i]+=c[i-1];
        for(int i=1;i<=tot;i++)A[--c[len[i]]]=i;

        for(int i=tot-1;i>=1;i--)T[fa[A[i]]]=merge(T[fa[A[i]]],T[A[i]]);
        for(int i=1;i<=tot;i++)Fa[i][0]=fa[i];
        for(int k=1;k<=19;k++)for(int i=1;i<=tot;i++)Fa[i][k]=Fa[Fa[i][k-1]][k-1];

    }

    inline void solve(int l,int r,int k){
        int u=rt[r],length=r-l+1;
        if(len[fa[u]]+1>length){
            for(int k=19;k>=0;k--)if(len[fa[Fa[u][k]]]+1>length)u=Fa[u][k];
            u=fa[u];
        }
        if(k<=sum[T[u]])printf("%d\n",query(T[u],1,n,k)-length+1);
        else printf("-1\n");
    }

}using namespace Suffix_Automaton;
int main()
{
    int T;
    scanf("%d",&T);
    while(T--){
        tot1=0;
        scanf("%d %d",&n,&q);
        scanf("%s",s+1);
        init(s,n);
        int l,r,k;
        while(q--){
            scanf("%d %d %d",&l,&r,&k);
            solve(l,r,k);
        }
    }
    return 0;
}
```

2.14 序列自动机

```
/*
用法类似后缀自动机
作用：识别一个串的子序列
构造复杂度：O(n^2)
*/
struct SqAM{
    int next[maxn<<1][26],pre[maxn<<1],lst[26];
    int root,tot;
    void init(){
        root=tot=1;
        for(int i=0;i<26;i++)lst[i]=1;
    }

    void extend(int c){
        int p=lst[c],np=++tot;
        pre[np]=p;
        for(int i=0;i<26;i++)
        for(int j=lst[i];j&& !next[j][c];j=pre[j])next[j][c]=np;
        lst[c]=np;
    }
};
```

2.15 回文自动机

```
//最长双倍回文串长度
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=5e5+5;

struct Palindromic_Automaton{
    //0偶根 1奇根 range[2-tot]
    int s[maxn<<1],now;
    int next[maxn<<1][26],fail[maxn<<1],len[maxn<<1],last,tot;
    int cnt[maxn<<1]; //状态i表示的回文串数目

    int trans[maxn<<1];

    void init(){
        s[0]=len[1]=-1;
        fail[0]=tot=now=1;
        last=len[0]=0;
        memset(next[0],0,sizeof(next[0]));
        memset(next[1],0,sizeof(next[1]));
    }
    int newnode(){
        tot++;
        memset(next[tot],0,sizeof(next[tot]));
        fail[tot]=cnt[tot]=len[tot]=0;
        return tot;
    }
    int getfail(int x){
        while(s[now-len[x]-2]!=s[now-1])x=fail[x];
        return x;
    }
    void extend(int c){
        s[now++]=c;
        int cur=getfail(last);
        if(!next[cur][c]){
            int p=newnode();len[p]=len[cur]+2;
            fail[p]=next[getfail(fail[cur])][c];
            next[cur][c]=p;

            if(len[p]<=2)trans[p]=fail[p];
            else{
                int tmp=trans[cur];
                while(s[now-len[tmp]-2] != s[now-1] || (len[tmp]+2)*2>len[p])tmp=fail[tmp];
                trans[p]=next[tmp][c];
            }
        }
        last=next[cur][c];
        cnt[last]++;
    }
    int count(){return tot-1;}
    void calc(){
        for(int i=tot;i>=2;--i) cnt[fail[i]]+=cnt[i];
        cnt[0]=cnt[1]=0;
    }
    int getans(){
        int ans=0;
        for(int i=2;i<=tot;i++){
            if(len[i]>ans && len[trans[i]]*2==len[i] && len[trans[i]]%2==0)ans=len[i];
        }
        return ans;
    }
}pam;

char t[maxn];
```

```
int main()
{
    int n;
    scanf("%d",&n);
    scanf("%s",t);
    pam.init();
    for(int i=0;i<n;++i){
        pam.extend(t[i]-'a');
    }
    printf("%d\n",pam.getans());
    return 0;
}
```

3 数据结构

3.1 树状数组

3.1.1 前缀和

```
int a[32005],c[32005];
int lowbit(int x){return x&(-x);}
int getsum(int x){int ans=0;for(;x>0;x-=lowbit(x))ans+=c[x];return ans;}
void upd(int x,int del){for(;x<32004;x+=lowbit(x))c[x]+=del;}
```

3.1.2 第 k 大

```
ll kth_element(ll k){
    ll ans=0,cnt=0;
    for(int i=16;i>=0;i--){
        ans+=(1LL<<i);
        if(ans>=maxn-5 || cnt+c[ans]>=k)ans-=(1<<i);
        elsecnt+=c[ans];
    }
    ans++;
    if(vis[ans])return ans;
    else return -1;
}
```

3.1.3 二维树状数组

```
//高维同理
int a[maxn][maxn],c[maxn][maxn];
int n,m;
int lowbit(int x){return x&(-x);}
long long getsum(int x,int y){
    long long ans=0;
    for(int i=x;i>0;i-=lowbit(i))
    for(int j=y;j>0;j-=lowbit(j))
        ans+=c[i][j];
    return ans%MOD;
}
void upd(int x,int y,long del){
    for(int i=x;i<=n;i+=lowbit(i))
    for(int j=y;j<=n;j+=lowbit(j))
    c[i][j]+=del;
}
```

3.2 线段树

3.2.1 动态开点

```
namespace Dynamic_Segment_Tree{
    int cnt;
    int T[55],lson[maxn<<5],rson[maxn<<5],mi[maxn<<5];
    void update(int &x,int l,int r,int p,int v){
        if(!x){
            x=++cnt; mi[x]=v;
            lson[x]=rson[x]=0;
        }
        mi[x]=min(mi[x],v);
        if(l==r)return;
        int mid=(l+r)>>1;
        if(p<=mid)update(lson[x],l,mid,p,v);
        else update(rson[x],mid+1,r,p,v);
    }
    int getmin(int x,int l,int r,int L,int R,int d){
        if(!x)return INF;
        if(l>=L && r<=R)return mi[x];
        int mid=(l+r)>>1;
        int res=INF;
        if(L<=mid)res=min(res,getmin(lson[x],l,mid,L,R,d));
        if(R>mid)res=min(res,getmin(rson[x],mid+1,r,L,R,d));
        return res;
    }
}using namespace Dynamic_Segment_Tree;
```

3.2.2 势能线段树

```
//区间取min,max
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=5e5+5;
const int INF=INT_MAX;
int n,m,a[maxn];

namespace Segment_Tree_Beats{
    #define N maxn<<2
    #define lc (x<<1)
    #define rc (x<<1|1)
    #define mid ((l+r)>>1)
    #define lson lc,l,mid
    #define rson rc,mid+1,r
    int ma[N],sma[N],cma[N];
    int mi[N],smi[N],cmi[N];
    int add[N],tma[N],tmi[N];
    ll sum[N];
    void push_add(int x,int l,int r,int val){
        sum[x]+=1ll*(r-l+1)*val;
        ma[x]+=val; mi[x]+=val; add[x]+=val;
        if(sma[x]!=-INF)sma[x]+=val;
        if(smi[x]!=INF)smi[x]+=val;
        if(tma[x]!=-INF)tma[x]+=val;
        if(tmi[x]!=INF)tmi[x]+=val;
    }
    void push_min(int x,int val){
        if(ma[x]<=val)return;
        sum[x]-=1ll*(ma[x]-val)*cma[x];
        if(mi[x]==ma[x])mi[x]=val;
        if(smi[x]==ma[x])smi[x]=val;
        if(tma[x]>val)tma[x]=val;
        ma[x]=tmi[x]=val;//positon can't change
    }
    void push_max(int x,int val){
        if(mi[x]>val)return;
        sum[x]+=1ll*(val-mi[x])*cmi[x];
        if(ma[x]==mi[x])ma[x]=val;
        if(sma[x]==mi[x])sma[x]=val;
        if(tmi[x]<val)tmi[x]=val;
        mi[x]=tma[x]=val;//positon can't change
    }
    void push_up(int x){
        sum[x]=sum[lc]+sum[rc];

        if(ma[lc]==ma[rc]){
            ma[x]=ma[lc]; cma[x]=cma[lc]+cma[rc];
            sma[x]=max(sma[lc],sma[rc]);
        }
        else if(ma[lc]>ma[rc]){
            ma[x]=ma[lc]; cma[x]=cma[lc];
            sma[x]=max(sma[lc],ma[rc]);
        }
        else{
            ma[x]=ma[rc]; cma[x]=cma[rc];
            sma[x]=max(sma[rc],ma[lc]);
        }

        if(mi[lc]==mi[rc]){
            mi[x]=mi[lc]; cmi[x]=cmi[lc]+cmi[rc];
            smi[x]=min(smi[lc],smi[rc]);
        }
        else if(mi[lc]<mi[rc]){
            mi[x]=mi[lc]; cmi[x]=cmi[lc];
            smi[x]=min(smi[lc],mi[rc]);
        }
        else{
            mi[x]=mi[rc]; cmi[x]=cmi[rc];
            smi[x]=min(smi[rc],mi[lc]);
        }
    }
    void push_down(int x,int l,int r){
        if(add[x]!=0){
            push_add(lc,l,mid,add[x]);
            push_add(rc,mid+1,r,add[x]);
            add[x]=0;
        }
        if(tma[x]!=-INF){
            push_max(lc,tma[x]);
            push_max(rc,tma[x]);
            tma[x]=-INF;
        }
        if(tmi[x]!=INF){
            push_min(lc,tmi[x]);
            push_min(rc,tmi[x]);
            tmi[x]=INF;
        }
    }
    void build(int x,int l,int r){
        tma[x]=-INF; tmi[x]=INF; add[x]=0;
        if(l==r){
            ma[x]=mi[x]=sum[x]=a[l];
            cma[x]=cmi[x]=1;
            sma[x]=-INF;smi[x]=INF;
            return;
        }
        build(lson);build(rson);
        push_up(x);
    }
    void modify_add(int x,int l,int r,int L,int R,int t){
        if(l>R || r<L)return;
        if(l>=L && r<=R){
            push_add(x,l,r,t);
            return;
        }
        push_down(x,l,r);
        modify_add(lson,L,R,t);
        modify_add(rson,L,R,t);
        push_up(x);
    }
    void modify_min(int x,int l,int r,int L,int R,int t){
        if(l>R || r<L || ma[x]<=t)return;
        if(l>=L && r<=R && sma[x]<t){
            push_min(x,t);
            return;
        }
        push_down(x,l,r);
        modify_min(lson,L,R,t);
        modify_min(rson,L,R,t);
        push_up(x);
    }
    void modify_max(int x,int l,int r,int L,int R,int t){
        if(l>R || r<L || mi[x]>=t)return;
        if(l>=L && r<=R && smi[x]>t){
            push_max(x,t);
            return;
        }
        push_down(x,l,r);
        modify_max(lson,L,R,t);
        modify_max(rson,L,R,t);
        push_up(x);
    }
    int query_max(int x,int l,int r,int L,int R){
        if(l>R || r<L)return -INF;
        if(l>=L && r<=R)return ma[x];
        push_down(x,l,r);
        return max(query_max(lson,L,R),query_max(rson,L,R));
    }
    int query_min(int x,int l,int r,int L,int R){
        if(l>R || r<L)return INF;
        if(l>=L && r<=R)return mi[x];
        push_down(x,l,r);
        return min(query_min(lson,L,R),query_min(rson,L,R));
    }
    ll query_sum(int x,int l,int r,int L,int R){
        if(l>R || r<L)return 0;
        if(l>=L && r<=R)return sum[x];
        push_down(x,l,r);
        return query_sum(lson,L,R)+query_sum(rson,L,R);
    }
}using namespace Segment_Tree_Beats;

int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);
    build(1,1,n);
    scanf("%d",&m);
    int op,l,r,x;
    while(m--){
        scanf("%d %d %d",&op,&l,&r);
        switch(op){
            case 1:scanf("%d",&x);modify_add(1,1,n,l,r,x);break;
            case 2:scanf("%d",&x);modify_max(1,1,n,l,r,x);break;
            case 3:scanf("%d",&x);modify_min(1,1,n,l,r,x);break;
            case 4:printf("%lld\n",query_sum(1,1,n,l,r));break;
            case 5:printf("% d\n",query_max(1,1,n,l,r));break;
            case 6:printf("%d\n",query_min(1,1,n,l,r));break;
        }
    }
    return 0;
}
```

3.2.3 区间历史最值

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=1e5+5;
const int INF=INT_MAX;

int n,m,a[maxn];

namespace Segment_Tree{
    const int N=maxn<<2;
    #define lc (x<<1)
    #define rc (x<<1|1)
    #define mid ((l+r)>>1)
    #define lson lc,l,mid
    #define rson rc,mid+1,r
    int nma[N],nadd[N],nset[N];//n-now
    int pma[N],padd[N],pset[N];//p-pre

    void push_padd(int x,int val){
        pma[x]=max(pma[x],nma[x]+val);
        if(nset[x]!=-INF)pset[x]=max(pset[x],nset[x]+val);
        else padd[x]=max(padd[x],nadd[x]+val);
    }

    void push_pset(int x,int val){
        pma[x]=max(pma[x],val);
        pset[x]=max(pset[x],val);
    }

    void push_nadd(int x,int val){
        nma[x]+=val; pma[x]=max(pma[x],nma[x]);
        if(nset[x]!=-INF)nset[x]+=val,pset[x]=max(pset[x],nset[x]);
        else nadd[x]+=val,padd[x]=max(padd[x],nadd[x]);
    }

    void push_nset(int x,int val){
        nma[x]=val; pma[x]=max(pma[x],nma[x]);
        nset[x]=val; pset[x]=max(pset[x],nset[x]);
        nadd[x]=0;
    }

    void push_up(int x){
        nma[x]=max(nma[lc],nma[rc]);
        pma[x]=max(pma[lc],pma[rc]);
    }

    void push_down(int x){
        if(padd[x]){
            push_padd(lc,padd[x]); push_padd(rc,padd[x]);
            padd[x]=0;
        }
        if(pset[x]!=-INF){
            push_pset(lc,pset[x]); push_pset(rc,pset[x]);
            pset[x]=-INF;
        }
```

```
      if(nadd[x]){
         push_nadd(lc,nadd[x]); push_nadd(rc,nadd[x]);
         nadd[x]=0;
      }
      if(nset[x]!=-INF){
         push_nset(lc,nset[x]); push_nset(rc,nset[x]);
         nset[x]=-INF;
      }
   }

   void build(int x,int l,int r){
      nset[x]=pset[x]=-INF;
      nadd[x]=padd[x]=0;
      if(l==r){
         nma[x]=pma[x]=a[l];
         return;
      }
      push_down(x);
      build(lson); build(rson);
      push_up(x);
   }

   void modify_add(int x,int l,int r,int L,int R,int d){
      if(l>R || r<L)return;
      if(l>=L && r<=R){
         push_nadd(x,d);
         return;
      }
      push_down(x);
      modify_add(lson,L,R,d);
      modify_add(rson,L,R,d);
      push_up(x);
   }

   void modify_set(int x,int l,int r,int L,int R,int d){
      if(l>R || r<L)return;
      if(l>=L && r<=R){
         push_nset(x,d);
         return;
      }
      push_down(x);
      modify_set(lson,L,R,d);
      modify_set(rson,L,R,d);
      push_up(x);
   }

   int query_nma(int x,int l,int r,int L,int R){
      if(l>R || r<L)return -INF;
      if(l>=L && r<=R)return nma[x];
      push_down(x);
      return max(query_nma(lson,L,R),query_nma(rson,L,R));
   }

   int query_pma(int x,int l,int r,int L,int R){
      if(l>R || r<L)return -INF;
      if(l>=L && r<=R)return pma[x];
      push_down(x);
      return max(query_pma(lson,L,R),query_pma(rson,L,R));
   }
}using namespace Segment_Tree;

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

```
    build(1,1,n);
    scanf("%d",&m);
    char op[5];
    int x,y,z;
    while(m--){
        scanf("%s%d%d",op,&x,&y);
        switch(op[0]){
            case 'Q':printf("%d\n",query_nma(1,1,n,x,y));break;
            case 'A':printf("%d\n",query_pma(1,1,n,x,y));break;
            case 'P':scanf("%d",&z);modify_add(1,1,n,x,y,z);break;
            case 'C':scanf("%d",&z);modify_set(1,1,n,x,y,z);break;
        }
    }
    return 0;
}
```

3.3 扫描线周长

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN=10010;
struct Node
{
    int l,r;
    int cnt;//有效长度
    int lf,rf;//实际的左右端点
    int numseg;//分支数，一个分支对应两条竖线
    int c;//记录覆盖情况
    bool lcover,rcover;
}segTree[MAXN*4];
struct Line
{
    int y;
    int x1,x2;
    int f;
}line[MAXN];
bool cmp(Line a,Line b)
{
    return a.y<b.y;
}
int x[MAXN];
void Build(int i,int l,int r)
{
    segTree[i].l=l; segTree[i].r=r;
    segTree[i].lf=x[l]; segTree[i].rf=x[r];
    segTree[i].cnt=0;
    segTree[i].numseg=0;
    segTree[i].c=0;
    segTree[i].lcover=segTree[i].rcover=false;
    if(l+1==r)return;
    int mid=(l+r)/2;
    Build(i<<1,l,mid);
    Build((i<<1)|1,mid,r);
}
void push_up(int i)
{
    if(segTree[i].c>0)
    {
        segTree[i].cnt=segTree[i].rf-segTree[i].lf;
        segTree[i].numseg=1;
        segTree[i].lcover=segTree[i].rcover=true;
        return;
    }
    if(segTree[i].l+1==segTree[i].r)
    {
        segTree[i].cnt=0;
        segTree[i].numseg=0;
        segTree[i].lcover=segTree[i].rcover=false;
    }
    else
    {
        segTree[i].cnt=segTree[i<<1].cnt+segTree[(i<<1)|1].cnt;
        segTree[i].lcover=segTree[i<<1].lcover;
        segTree[i].rcover=segTree[(i<<1)|1].rcover;
        segTree[i].numseg=segTree[i<<1].numseg+segTree[(i<<1)|1].numseg;
        if(segTree[i<<1].rcover&&segTree[(i<<1)|1].lcover)segTree[i].numseg--;
    }
}
void update(int i,Line e)
{
    if(segTree[i].lf==e.x1&&segTree[i].rf==e.x2)
    {
        segTree[i].c+=e.f;
        push_up(i);
        return;
    }
    if(e.x2<=segTree[i<<1].rf)update(i<<1,e);
    else if(e.x1>=segTree[(i<<1)|1].lf)update((i<<1)|1,e);
    else
    {
        Line temp=e;
        temp.x2=segTree[i<<1].rf;
        update(i<<1,temp);
        temp=e;
        temp.x1=segTree[(i<<1)|1].lf;
        update((i<<1)|1,temp);
    }
    push_up(i);
}
int main()
{
    int x1,y1,x2,y2;
    int n;
    while(~scanf("%d",&n))
    {
        int t=0;
        for(int i=0;i<n;i++)
        {
            scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            line[t].x1=x1;line[t].x2=x2;
            line[t].y=y1;line[t].f=1;
            x[t++]=x1;

            line[t].x1=x1;line[t].x2=x2;
            line[t].y=y2;line[t].f=-1;
            x[t++]=x2;
        }
        sort(line,line+t,cmp);

        sort(x,x+t);
        int m=unique(x,x+t)-x;
        Build(1,0,m-1);
        int ans=0;
        int last=0;
        for(int i=0;i<t-1;i++)
        {
            update(1,line[i]);
            ans+=segTree[1].numseg*2*(line[i+1].y-line[i].y);
            ans+=abs(segTree[1].cnt-last);
            last=segTree[1].cnt;
        }
        update(1,line[t-1]);
        ans+=abs(segTree[1].cnt-last);
        printf("%d\n",ans);
    }
    return 0;
}
```

3.4 扫描线面积

```
//tips:维护覆盖的区间长度，标记lazy tag,若整个区间都被覆盖一次，就记1,并设置sum为区间长度
//若区间未被全部覆盖则合并左右子树的信息
//hdu 1255:至少覆盖2次的面积
#include <bits/stdc++.h>
#define lc idx<<1
#define rc idx<<1|1
#define lson l,mid,lc
#define rson mid,r,rc
#define N 2222

using namespace std;
int n,m;
double X[N];

struct line {
    double lx,rx,y;
    int flag;
} a[N];

struct Tree {
    double len;
    double one;
    int cnt;
} tree[N<<2];

bool cmp(line a,line b) {
    if(a.y==b.y)return a.flag>b.flag;
    return a.y<b.y;
}

void push_up(int l,int r,int idx) {
    if(tree[idx].cnt>=2) {
        tree[idx].len=X[r]-X[l];
        tree[idx].one=0;
    } else if(tree[idx].cnt==1) {
        if(l+1==r)tree[idx].len=0;
        else {
            tree[idx].len=tree[lc].len+tree[rc].len+tree[lc].one+tree[rc].one;
        }
        tree[idx].one=X[r]-X[l]-tree[idx].len;
    } else {
        if(l+1==r)tree[idx].len=tree[idx].one=0;
        else {
            tree[idx].len=tree[lc].len+tree[rc].len;
            tree[idx].one=tree[lc].one+tree[rc].one;
        }
    }
}

void build(int l,int r,int idx) {
    tree[idx].len=tree[idx].one=0;
    tree[idx].cnt=0;
    if(l+1==r)return;
    int mid=(l+r)>>1;
    build(lson);
    build(rson);
}

void update(int l,int r,int idx,int x,int y,int flag) {
    if(x<=l&&r<=y) {
        tree[idx].cnt+=flag;
        push_up(l,r,idx);
        return;
    }
    int mid=(l+r)>>1;
    if(x<mid)update(lson,x,y,flag);
    if(y>mid)update(rson,x,y,flag);
    push_up(l,r,idx);
}

int main() {
    int t;
    cin>>t;
    while(t--) {
        double x,y,_x,_y;
        scanf("%d",&n);
        m=1;
        for(int i=0; i<n; i++) {
            scanf("%lf%lf%lf%lf",&x,&y,&_x,&_y);
            X[m]=x;
            a[m].lx=x,a[m].rx=_x;
            a[m].y=y,a[m++].flag=1;
            X[m]=_x;
            a[m].lx=x,a[m].rx=_x;
            a[m].y=_y,a[m++].flag=-1;
        }
        sort(X+1,X+m);
        sort(a+1,a+m,cmp);
        int mm=1;
        X[mm++]=X[1];
        for(int i=2; i<m; i++) {
            if(X[i]!=X[i-1])X[mm++]=X[i];
        }
        build(1,mm-1,1);
        double ans=0;
        for(int i=1; i<m-1; i++) {
            int l=lower_bound(X+1,X+mm,a[i].lx)-X;
            int r=lower_bound(X+1,X+mm,a[i].rx)-X;
            update(1,mm-1,1,l,r,a[i].flag);
            ans+=tree[1].len*(a[i+1].y-a[i].y);
        }
        printf("%.2f\n",ans);
    }
    return 0;
}
```

3.5 主席树

3.5.1 静态第 k 小

```
#include <bits/stdc++.h>
#define mid (l+r)/2
using namespace std;

const int N = 200010;
int n, q, m, cnt = 0;
int a[N], b[N], T[N];
int sum[N<<5], L[N<<5], R[N<<5];

inline int build(int l, int r){
    int rt = ++ cnt;
    sum[rt] = 0;
    if (l < r){
        L[rt] = build(l, mid);
        R[rt] = build(mid+1, r);
    }
    return rt;
}

inline int update(int pre, int l, int r, int x){
    int rt = ++ cnt;
    L[rt] = L[pre]; R[rt] = R[pre]; sum[rt] = sum[pre]+1;
    if (l < r){
        if (x <= mid) L[rt] = update(L[pre], l, mid, x);
        else R[rt] = update(R[pre], mid+1, r, x);
    }
    return rt;
}

inline int query(int u, int v, int l, int r, int k){
    if (l >= r) return l;
    int x = sum[L[v]] - sum[L[u]];
    if (x >= k) return query(L[u], L[v], l, mid, k);
    else return query(R[u], R[v], mid+1, r, k-x);
}

int main(){
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i ++){
        scanf("%d", &a[i]);
        b[i] = a[i];
    }
    sort(b+1, b+1+n);
    m = unique(b+1, b+1+n)-b-1;
    T[0] = build(1, m);
    for (int i = 1; i <= n; i ++){
        int t = lower_bound(b+1, b+1+m, a[i])-b;
        T[i] = update(T[i-1], 1, m, t);
    }
    while (q --){
        int x, y, z;
        scanf("%d%d%d", &x, &y, &z);
        int t = query(T[x-1], T[y], 1, m, z);
        printf("%d\n", b[t]);
    }
    return 0;
}
```

3.5.2 带修改第 k 小

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 60010;
const int M = 2500010;
int n,q,m,tot;
int a[MAXN], t[MAXN];
int T[MAXN], lson[M], rson[M],c[M];
int S[MAXN];

struct Query{
    int kind;
    int l,r,k;
}query[10010];
```

```
void Init_hash(int k){
    sort(t,t+k);
    m = unique(t,t+k) - t;
}
int hash(int x){return lower_bound(t,t+m,x)-t;}
int build(int l,int r){
    int root = tot++;
    c[root] = 0;
    if(l != r){
        int mid = (l+r)/2;
        lson[root] = build(l,mid);
        rson[root] = build(mid+1,r);
    }
    return root;
}

int Insert(int root,int pos,int val){
    int newroot = tot++, tmp = newroot;
    int l = 0, r = m-1;
    c[newroot] = c[root] + val;
    while(l < r){
        int mid = (l+r)>>1;
        if(pos <= mid){
            lson[newroot] = tot++; rson[newroot] = rson[root];
            newroot = lson[newroot]; root = lson[root];
            r = mid;
        }
        else{
            rson[newroot] = tot++; lson[newroot] = lson[root];
            newroot = rson[newroot]; root = rson[root];
            l = mid+1;
        }
        c[newroot] = c[root] + val;
    }
    return tmp;
}

int lowbit(int x){return x&(-x);}
int use[MAXN];
void add(int x,int pos,int val){
    while(x <= n){
        S[x] = Insert(S[x],pos,val);
        x += lowbit(x);
    }
}
int sum(int x){
    int ret = 0;
    while(x > 0){
        ret += c[lson[use[x]]];
        x -= lowbit(x);
    }
    return ret;
}
int Query(int left,int right,int k)
{
    int left_root = T[left-1];
    int right_root = T[right];
    int l = 0, r = m-1;
    for(int i = left-1;i;i -= lowbit(i)) use[i] = S[i];
    for(int i = right;i ;i -= lowbit(i)) use[i] = S[i];
    while(l < r){
        int mid = (l+r)/2;
        int tmp = sum(right) - sum(left-1) + c[lson[right_root]] - c[lson[left_root]];
        if(tmp >= k){
            r = mid;
            for(int i = left-1; i ;i -= lowbit(i))use[i] = lson[use[i]];
            for(int i = right; i; i -= lowbit(i))use[i] = lson[use[i]];
            left_root = lson[left_root];
            right_root = lson[right_root];
        }
        else{
            l = mid+1; k -= tmp;
            for(int i = left-1; i;i -= lowbit(i))use[i] = rson[use[i]];
            for(int i = right;i ;i -= lowbit(i))use[i] = rson[use[i]];
            left_root = rson[left_root];
            right_root = rson[right_root];
        }
    }
    return l;
}
void Modify(int x,int p,int d){
    while(x <= n){
        S[x] = Insert(S[x],p,d);
        x += lowbit(x);
    }
}

int main(){
    int Tcase;
    scanf("%d",&Tcase);
    while(Tcase--){
        scanf("%d%d",&n,&q);
        tot = 0; m = 0;
        for(int i = 1;i <= n;i++){
            scanf("%d",&a[i]);
            t[m++] = a[i];
        }
        char op[10];
        for(int i = 0;i < q;i++){
            scanf("%s",op);
            if(op[0] == 'Q'){
                query[i].kind = 0;
                scanf("%d%d%d",&query[i].l,&query[i].r,&query[i].k);
            }
            else{
                query[i].kind = 1;
                scanf("%d%d",&query[i].l,&query[i].r);
                t[m++] = query[i].r;
            }
        }
        Init_hash(m);
        T[0] = build(0,m-1);
        for(int i = 1;i <= n;i++)
        T[i] = Insert(T[i-1],hash(a[i]),1);
        for(int i = 1;i <= n;i++)
        S[i] = T[0];
        for(int i = 0;i < q;i++){
            if(query[i].kind == 0)
            printf("%d\n",t[Query(query[i].l,query[i].r,query[i].k)]);
            else{
                Modify(query[i].l,hash(a[query[i].l]),-1);
                Modify(query[i].l,hash(query[i].r),1);
                a[query[i].l] = query[i].r;
            }
        }
    }
    return 0;
}
```

3.5.3 封装模板

```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int maxn=1e5+5;

int n,m,q;
int a[maxn];
int T[maxn],sz,cnt;
int nt[maxn];

struct persist_segtree{
   struct node{
      int lft,rgt;
      ll num,lazy;
   }segs[maxn*26];
   int tot;

   void init(){
      tot=0;
      memset(segs,0,sizeof(segs));
   }

   inline ll getnum(int rt){
      return segs[rt].num;
   }

   void up(int rt){
      segs[rt].num=getnum(segs[rt].lft)+getnum(segs[rt].rgt);
   }

   void down(int rt,int lft,int rgt){
      segs[rt].num+=segs[rt].lazy*(rgt-lft+1);

      int mid=(lft+rgt)/2;
      segs[++tot]=segs[segs[rt].lft];
      segs[tot].lazy+=segs[rt].lazy;
      segs[rt].lft=tot;

      segs[++tot]=segs[segs[rt].rgt];
      segs[tot].lazy+=segs[rt].lazy;
      segs[rt].rgt=tot;

      segs[rt].lazy=0;
   }

   void build(int &rt,int l,int r){
      rt=++tot;
      if(l!=r){
         int mid=(l+r)/2;
         build(segs[rt].lft,l,mid);
         build(segs[rt].rgt,mid+1,r);
         up(rt);
      }
      else segs[rt].num=a[++cnt];
   }

   int add(int &rt,int lft,int rgt,int beg,int end,int num){
      //cout<<"Add:"<<lft<<" "<<rgt<<" "<<beg<<" "<<end<<" "<<num<<endl;
      int nrt=++tot;
      segs[nrt]=segs[rt];
      rt=nrt;
      if(beg==lft && end==rgt){
            segs[rt].lazy+=num;
            return nrt;
        }
        segs[rt].num+=1LL*num*(end-beg+1);
        //if(segs[rt].lazy)down(rt,lft,rgt);
        int mid=(lft+rgt)/2;
        if(end<=mid)add(segs[rt].lft,lft,mid,beg,end,num);
        else if(beg>mid)add(segs[rt].rgt,mid+1,rgt,beg,end,num);
        else{
            add(segs[rt].lft,lft,mid,beg,mid,num);
            add(segs[rt].rgt,mid+1,rgt,mid+1,end,num);
        }
        return nrt;
    }

    ll query(int rt,int lft,int rgt,int beg,int end){
        //cout<<"Query:"<<lft<<" "<<rgt<<" "<<beg<<" "<<end<<endl;
        if(lft==beg && rgt==end)return segs[rt].num+segs[rt].lazy*(end-beg+1);
        int mid=(lft+rgt)/2;
        if(end<=mid)return segs[rt].lazy*(end-beg+1)+query(segs[rt].lft,lft,mid,beg,end);
        else if(beg>mid)return segs[rt].lazy*(end-beg+1)+query(segs[rt].rgt,mid+1,rgt,beg,end);
        return segs[rt].lazy*(end-beg+1)+query(segs[rt].lft,lft,mid,beg,mid)+query(segs[rt].rgt,mid+1,rgt,mid+1,
            end);
    }
}PST;

int main()
{
    //freopen("in.txt","r",stdin);
    while(~scanf("%d %d",&n,&q)){
        sz=0;cnt=0;
        for(ll i=1;i<=n;i++)scanf("%d",&a[i]);
        PST.init();
        PST.build(T[0],1,n);
        nt[0]=PST.tot;
        char cmd[10];
        int l,r,t,d;
        ll ans;
        while(q--){
            scanf("%s",cmd);
            switch(cmd[0])
            {
                case 'C':
                    scanf("%d %d %d",&l,&r,&d);
                    ++sz;
                    T[sz]=T[sz-1];
                    PST.add(T[sz],1,n,l,r,d);
                    nt[sz]=PST.tot;
                    break;
                case 'Q':
                    scanf("%d %d",&l,&r);
                    ans=PST.query(T[sz],1,n,l,r);
                    printf("%lld\n",ans);
                    break;
                case 'H':
                    scanf("%d %d %d",&l,&r,&t);
                    ans=PST.query(T[t],1,n,l,r);
                    printf("%lld\n",ans);
                    break;
                case 'B':
                    scanf("%d",&t);
                    sz=t;
                    PST.tot=nt[sz];
                    break;
            }
            //cout<<"Debug:";
            //for(int i=1;i<=n;i++)cout<<PST.query(T[sz],1,n,i,i)<<" ";cout<<endl;
        }
    }
    return 0;
}
```

3.6 平衡树

3.6.1 权值 Splay

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=1e6+5;
const int INF=1e9+7;

int ch[maxn][2],f[maxn],sz[maxn],cnt[maxn],val[maxn];
int tot,root;

inline void init(){tot=root=0;}

inline void clear(int x){ch[x][0]=ch[x][1]=f[x]=sz[x]=cnt[x]=val[x]=0;}

inline bool get(int x){return ch[f[x]][1]==x;}

inline void push_up(int x)
{
    if(!x)return;
    sz[x]=cnt[x];
    if(ch[x][0])sz[x]+=sz[ch[x][0]];
    if(ch[x][1])sz[x]+=sz[ch[x][1]];
}

inline void rotate(int x)
{
    int old=f[x],oldf=f[old],whichx=get(x);
    ch[old][whichx]=ch[x][whichx^1]; f[ch[old][whichx]]=old;
    ch[x][whichx^1]=old; f[old]=x;
    f[x]=oldf;
    if(oldf)ch[oldf][ch[oldf][1]==old]=x;
    push_up(old); push_up(x);
}

inline void splay(int x,int ed)
{
    for(int fa;(fa=f[x])!=ed;rotate(x))
    if(f[fa]!=ed)rotate(((get(x)==get(fa))?fa:x));
    if(ed==0)root=x;
}

inline void insert(int x)
{
    if(root==0)
    {
        tot++; ch[tot][0]=ch[tot][1]=f[tot]=0;
        root=tot; sz[tot]=cnt[tot]=1; val[tot]=x;
        return;
    }
    int now=root,fa=0;
    while(true)
    {
        if(x==val[now])
        {
            cnt[now]++;
            push_up(now); push_up(fa);
            splay(now,0);
            break;
        }
        fa=now;
        now=ch[now][val[now]<x];
        if(now==0)
        {
            tot++;
            ch[tot][0]=ch[tot][1]=0; f[tot]=fa;
            sz[tot]=cnt[tot]=1;
            ch[fa][val[fa]<x]=tot;
            val[tot]=x;
            push_up(fa); splay(tot,0);
            break;
        }
    }
}

inline int getrank(int now,int x)
{
    int ans=0;
    while(true)
    {
        if(x<val[now])now=ch[now][0];
        else
        {
            ans+=(ch[now][0]?sz[ch[now][0]]:0);
            if(x==val[now]){splay(now,0);return ans+1;}
            ans+=cnt[now];
            now=ch[now][1];
        }
    }
}

inline int getkth(int now,int k)
{
    while(now)
    {
        if(k<=sz[ch[now][0]]) now=ch[now][0];
        else if(k<=sz[ch[now][0]]+cnt[now])return val[now];
        else k-=sz[ch[now][0]]+cnt[now],now=ch[now][1];
    }
    return 0;
}

inline int getpre(int now)
{
    now=ch[now][0];
    while(ch[now][1])now=ch[now][1];
    return now;
}

inline int getsuc(int now)
{
    now=ch[now][1];
    while(ch[now][0])now=ch[now][0];
    return now;
}

inline void del(int x)
{
    int whatever=getrank(root,x);
    if (cnt[root]>1){cnt[root]--; push_up(root); return;}
    if (!ch[root][0]&&!ch[root][1]) {clear(root); root=0; return;}
    if (!ch[root][0])
    {
        int oldroot=root; root=ch[root][1]; f[root]=0; clear(oldroot); return;
    }
    else if (!ch[root][1])
    {
        int oldroot=root; root=ch[root][0]; f[root]=0; clear(oldroot); return;
    }
    int leftbig=getpre(root),oldroot=root;
    splay(leftbig,0);
    ch[root][1]=ch[oldroot][1];
    f[ch[oldroot][1]]=root;
    clear(oldroot);
    push_up(root);
}

int n;

int main()
{
    init();
    scanf("%d",&n);
    int op,v;
    for(int i=1;i<=n;++i)
    {
        scanf("%d %d",&op,&v);
        switch(op)
        {
            case 1:insert(v);break;
            case 2:del(v);break;
            case 3:printf("%d\n",getrank(root,v));break;
            case 4:printf("%d\n",getkth(root,v));break;
            case 5:insert(v);printf("%d\n",val[getpre(root)]);del(v);break;
            case 6:insert(v);printf("%d\n",val[getsuc(root)]);del(v);break;
        }
    }
    return 0;
}
```

3.6.2 区间 Splay

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=5e5+5;
const int INF=1e9+7;

int a[maxn];
queue<int>zzk;


int ch[maxn][2],f[maxn],sz[maxn],cnt[maxn],val[maxn];
int setv[maxn],rev[maxn],lm[maxn],rm[maxn],mx[maxn],sum[maxn];

int tot,root;

inline void init(){tot=root=0;}

inline void clear(int x)
{
    ch[x][0]=ch[x][1]=f[x]=sz[x]=cnt[x]=val[x]=0;
    setv[x]=INF; rev[x]=lm[x]=rm[x]=mx[x]=sum[x]=0;
}
```

```
void gc(int &x)
{
   if(!x)return;
   gc(ch[x][0]);gc(ch[x][1]);
   zzk.push(x);
   x=0;
}

inline bool get(int x){return ch[f[x]][1]==x;}

inline void push_up(int x)
{
   int l=ch[x][0],r=ch[x][1];
   sum[x]=sum[l]+sum[r]+val[x];
   sz[x]=sz[l]+sz[r]+1;
   mx[x]=max(mx[l],mx[r]);
   mx[x]=max(mx[x],rm[l]+val[x]+lm[r]);
   lm[x]=max(lm[l],sum[l]+val[x]+lm[r]);
   rm[x]=max(rm[r],rm[l]+val[x]+sum[r]);
}

inline void setval(int x,int v)
{
   if(!x)return;
   val[x]=v;
   sum[x]=sz[x]*val[x];
   lm[x]=rm[x]=max(0,sum[x]);
   mx[x]=max(sum[x],val[x]);
   setv[x]=v;
}

inline void reverse(int x)
{
   if(!x)return;
   swap(ch[x][0],ch[x][1]);
   swap(lm[x],rm[x]);
   rev[x]^=1;
}

inline void push_down(int x)
{
   if(setv[x]!=INF)
   {
      setval(ch[x][0],setv[x]);setval(ch[x][1],setv[x]);
      setv[x]=INF;
   }
   if(rev[x]!=0)
   {
      reverse(ch[x][0]);reverse(ch[x][1]);
      rev[x]=0;
   }
}

inline void rotate(int x)
{
   int old=f[x],oldf=f[old],whichx=get(x);
   ch[old][whichx]=ch[x][whichx^1]; f[ch[old][whichx]]=old;
   ch[x][whichx^1]=old; f[old]=x;
   f[x]=oldf;
   if(oldf)ch[oldf][ch[oldf][1]==old]=x;
   push_up(old); push_up(x);
}

inline void splay(int x,int ed)
```

```
{
   for(int fa;(fa=f[x])!=ed;rotate(x))
   if(f[fa]!=ed)rotate(((get(x)==get(fa))?fa:x));
   if(ed==0)root=x;
}

inline int kth(int now,int k)
{
   while(now)
   {
      push_down(now);
      if(k<=sz[ch[now][0]]) now=ch[now][0];
      else if(k<=sz[ch[now][0]]+cnt[now])return now;
      else k-=sz[ch[now][0]]+cnt[now],now=ch[now][1];
   }
   return 0;
}

int build(int l,int r,int fa)
{
   if(l>r)return 0;
   int mid=(l+r)/2;
   int now;

   if(zzk.empty())now=++tot;
   else now=zzk.front(),zzk.pop();

   ch[now][0]=ch[now][1]=0; f[now]=fa;
   mx[now]=sum[now]=val[now]=a[mid];
   cnt[now]=1;sz[now]=1;
   rev[now]=0;setv[now]=INF;
   lm[now]=rm[now]=max(0,sum[now]);


   ch[now][0]=build(l,mid-1,now);
   ch[now][1]=build(mid+1,r,now);

   push_up(now);
   return now;
}

void INSERT()
{
   int pos,n;
   scanf("%d %d",&pos,&n);
   for(int i=1;i<=n;i++)scanf("%d",&a[i]);
   int rt=build(1,n,0);

   int l=kth(root,pos+1),r=kth(root,pos+2);
   splay(l,0);splay(r,l);
   int p=ch[root][1];
   f[rt]=p;ch[p][0]=rt;

   push_up(rt);push_up(r);push_up(l);
}

void DELETE()
{
   int pos,n;
   scanf("%d %d",&pos,&n);

   int l=kth(root,pos),r=kth(root,pos+n+1);
   splay(l,0);splay(r,l);
   gc(ch[r][0]);
   push_up(r);push_up(l);
```

```
}

void MAKESAME()
{
    int pos,n,c;
    scanf("%d %d %d",&pos,&n,&c);

    int l=kth(root,pos),r=kth(root,pos+n+1);
    splay(l,0);splay(r,l);
    int p=ch[root][1];p=ch[p][0];
    setval(p,c);
    push_up(r);push_up(l);
}

void REVERSE()
{
    int pos,n;
    scanf("%d %d",&pos,&n);

    int l=kth(root,pos),r=kth(root,pos+n+1);
    splay(l,0);splay(r,l);
    int p=ch[root][1];p=ch[p][0];
    reverse(p);
    push_up(r);push_up(l);
}

void GETSUM()
{
    int pos,n;
    scanf("%d %d",&pos,&n);

    int l=kth(root,pos),r=kth(root,pos+n+1);
    splay(l,0);splay(r,l);
    int p=ch[root][1];p=ch[p][0];
    printf("%d\n",sum[p]);
}

void MAXSUM()
{
    printf("%d\n",mx[root]);
}



void dfs(int x)
{
    if(!x)return;
    push_down(x);
    dfs(ch[x][0]);
    if(val[x]!=-INF && val[x]!=INF)printf("%d ",val[x]);
    dfs(ch[x][1]);
}

void print()
{
    cout<<"CHECK:";
    dfs(root);
    cout<<endl;
}



int main()
{
    int n,m;
    scanf("%d %d",&n,&m);
    mx[0]=-INF;//attention!
    a[1]=-INF;
    for(int i=2;i<=n+1;i++)scanf("%d",&a[i]);
    a[n+2]=-INF;

    init();
    root=build(1,n+2,0);

    char op[10];
    for(int i=1;i<=m;i++){
        //print();
        scanf("%s",op);
        switch(op[0])
        {
            case 'I':INSERT();break;
            case 'D':DELETE();break;
            case 'R':REVERSE();break;
            case 'G':GETSUM();break;
            case 'M':
            if(op[2]=='K')MAKESAME();
            else MAXSUM();
            break;
        }
    }
    return 0;
}
```

3.6.3 权值 treap

```
/*
split完一定要merge
*/
namespace treap
{
    struct node
    {
        int l,r,v,rnd,sz;
    } tr[maxn];
    int tot,root;

    inline void init()
    {
        tot=0;
        root=0;
    }

    inline void update(int x)
    {
        tr[x].sz=tr[tr[x].l].sz+tr[tr[x].r].sz+1;
    }

    inline int newnode(int v)
    {
        ++tot;
        tr[tot].sz=1;tr[tot].v=v;tr[tot].rnd=rand();
        tr[tot].l=tr[tot].r=0;
        return tot;
    }

    int merge(int x,int y)//x<y
    {
        if(!x || !y)return x+y;
        if(tr[x].rnd<tr[y].rnd)
        {
```

```
        tr[x].r=merge(tr[x].r,y);
        update(x);
        return x;
    }
    else
    {
        tr[y].l=merge(x,tr[y].l);
        update(y);
        return y;
    }
}

void split(int now,int k,int &x,int &y){
    if(!now)x=y=0;
    else
    {
        if(tr[now].v<=k)
        {
            x=now;
            split(tr[now].r,k,tr[now].r,y);
        }
        else
        {
            y=now;
            split(tr[now].l,k,x,tr[now].l);
        }
        update(now);
    }
}

inline void Insert(int &rt,int v)
{
    int x,y;
    split(rt,v,x,y);
    rt=merge(merge(x,newnode(v)),y);
}

inline void Delete(int &rt,int v)
{
    int x1,y1,x2,y2;
    split(rt,v,x1,y1);
    split(x1,v-1,x2,y2);
    y2=merge(tr[y2].l,tr[y2].r);
    rt=merge(merge(x2,y2),y1);
}

inline int getrank(int &rt,int v)
{
    int x,y;
    split(rt,v-1,x,y);
    int res=tr[x].sz;
    rt=merge(x,y);
    return res;
}

inline int getpre(int &rt,int v)
{
    int x,y;
    split(rt,v-1,x,y);
    if(x==0)return -INF;
    int res=x;
    while(tr[res].r)res=tr[res].r;
    rt=merge(x,y);
    return tr[res].v;
}
```

```
    inline int getsuc(int &rt,int v)
    {
        int x,y;
        split(rt,v,x,y);
        if(y==0)return INF;
        int res=y;
        while(tr[res].l)res=tr[res].l;
        rt=merge(x,y);
        return tr[res].v;
    }
}
```

3.6.4 区间 treap

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=5e5+5;
const int INF=1e9+9;

int n,m;
int c[maxn];
queue<int>zzk;
int s[maxn],top;
/*
split完一定要merge
*/

struct node
{
    int l,r,rnd,sz;
    int val,sum,mx,lm,rm;
    int setv,rev;
}tr[maxn];
int tot,root;

void gc(int x){
    if(!x)return;
    zzk.push(x);
    gc(tr[x].l);gc(tr[x].r);
}

void setval(int x,int val){
    tr[x].val=val;
    tr[x].sum=val*tr[x].sz;
    tr[x].lm=tr[x].rm=max(0,tr[x].sum);
    tr[x].mx=max(tr[x].sum,tr[x].val);
    tr[x].setv=val;
}

void reverse(int x){
    swap(tr[x].l,tr[x].r);
    swap(tr[x].lm,tr[x].rm);
    tr[x].rev^=1;
}

int newnode(int v)
{
    int x=0;
    if(zzk.empty())x=++tot;
    else x=zzk.front(),zzk.pop();
    tr[x].sz=1;
    tr[x].rnd=rand();
```

```cpp
    tr[x].val=tr[x].sum=tr[x].mx=v;
    tr[x].lm=tr[x].rm=max(0,v);
    tr[x].setv=INF;tr[x].rev=0;
    tr[x].l=0;tr[x].r=0;
    return x;
}

void push_up(int x){
    if(!tr[x].l && !tr[x].r){
        tr[x].sz=1;
        tr[x].sum=tr[x].mx=tr[x].val;
        tr[x].lm=tr[x].rm=max(0,tr[x].val);
    }
    else if(!tr[x].l){
        tr[x].sz=tr[tr[x].r].sz+1;
        tr[x].sum=tr[tr[x].r].sum+tr[x].val;
        tr[x].mx=max(tr[tr[x].r].mx,tr[tr[x].r].lm+tr[x].val);
        tr[x].lm=max(0,tr[tr[x].r].lm+tr[x].val);
        tr[x].rm=max(tr[tr[x].r].rm,tr[tr[x].r].sum+tr[x].val);
    }
    else if(!tr[x].r){
        tr[x].sz=tr[tr[x].l].sz+1;
        tr[x].sum=tr[tr[x].l].sum+tr[x].val;
        tr[x].mx=max(tr[tr[x].l].mx,tr[tr[x].l].rm+tr[x].val);
        tr[x].lm=max(tr[tr[x].l].lm,tr[tr[x].l].sum+tr[x].val);
        tr[x].rm=max(0,tr[tr[x].l].rm+tr[x].val);
    }
    else{
        tr[x].sz=tr[tr[x].l].sz+tr[tr[x].r].sz+1;
        tr[x].sum=tr[tr[x].l].sum+tr[tr[x].r].sum+tr[x].val;
        tr[x].mx=max(tr[tr[x].l].mx,tr[tr[x].r].mx);
        tr[x].mx=max(tr[x].mx,tr[tr[x].l].rm+tr[x].val+tr[tr[x].r].lm);
        tr[x].lm=max(tr[tr[x].l].lm,tr[tr[x].l].sum+tr[x].val+tr[tr[x].r].lm);
        tr[x].rm=max(tr[tr[x].r].rm,tr[tr[x].l].rm+tr[x].val+tr[tr[x].r].sum);
    }
}

void push_down(int x)
{
    if(tr[x].setv!=INF){
        if(tr[x].l)setval(tr[x].l,tr[x].setv);
        if(tr[x].r)setval(tr[x].r,tr[x].setv);
    }
    if(tr[x].rev)
    {
        if(tr[x].l)reverse(tr[x].l);
        if(tr[x].r)reverse(tr[x].r);
    }
    tr[x].setv=INF;
    tr[x].rev=0;
}

//for (int i = 1; i <= n; i++) root = merge(root, new_node(i));
//暴力建树, O(nlogn)

//单调栈维护树的最右链, O(n)建树
int build(int *a,int n)
{
    int x,y;
    top=0;
    for(int i=1;i<=n;i++){
        x=newnode(a[i]);y=0;
        while(top && tr[x].rnd<tr[s[top]].rnd){y=s[top--];push_up(y);}
        if(top)tr[s[top]].r=x;
        tr[x].l=y;
        s[++top]=x;
    }
    while(top)push_up(s[top--]);
    return s[1];
}

void init(){
    tot=0;
    for(int i=1;i<=n;i++)scanf("%d",&c[i]);
    root=build(c,n);
}

int merge(int x,int y)//x<y
{
    if(x)push_down(x);
    if(y)push_down(y);
    if(!x || !y)return x+y;
    if(tr[x].rnd<tr[y].rnd)
    {
        tr[x].r=merge(tr[x].r,y);
        push_up(x);
        return x;
    }
    else
    {
        tr[y].l=merge(x,tr[y].l);
        push_up(y);
        return y;
    }
}

//按size分
void split(int now,int k,int &x,int &y){
    if(!now)x=y=0;
    else
    {
        push_down(now);
        if(tr[tr[now].l].sz>=k)
        {
            y=now;
            split(tr[now].l,k,x,tr[now].l);
        }
        else
        {
            x=now;
            split(tr[now].r,k-tr[tr[now].l].sz-1,tr[now].r,y);
        }
        push_up(now);
    }
}

void Insert()
{
    int pos,cnt,x,y;
    scanf("%d %d",&pos,&cnt);
    for(int i=1;i<=cnt;i++)scanf("%d",&c[i]);
    int rt=build(c,cnt);
    split(root,pos,x,y);
    root=merge(merge(x,rt),y);
}

void Delete()
{
    int pos,cnt,x,y,z,w;
    scanf("%d %d",&pos,&cnt);
    split(root,pos-1,x,y);
    split(y,cnt,w,z);
    root=merge(x,z);
    gc(w);
}

void Makesame()
{
    int pos,cnt,x,y,z,w,vv;
    scanf("%d %d %d",&pos,&cnt,&vv);
    split(root,pos-1,x,y);
    split(y,cnt,z,w);
    setval(z,vv);
    root=merge(x,merge(z,w));
}

void Reverse()
{
    int pos,cnt,x,y,z,w;
    scanf("%d %d",&pos,&cnt);
    split(root,pos-1,x,y);
    split(y,cnt,z,w);
    reverse(z);
    root=merge(x,merge(z,w));
}

void Getsum()
{
    int pos,cnt,x,y,z,w;
    scanf("%d %d",&pos,&cnt);
    split(root,pos-1,x,y);
    split(y,cnt,z,w);
    int ans=tr[z].sum;
    root=merge(x,merge(z,w));
    printf("%d\n",ans);
}

void Getmaxsum()
{
    int ans=tr[root].mx;
    printf("%d\n",ans);
}

void getidx(int idx){
    int x,y,z,w;
    split(root,idx-1,x,y);
    split(y,1,z,w);
    int ans=tr[z].val;
    root=merge(x,merge(z,w));
    printf("%d ",ans);
}

void check()
{
    cout<<"CHECK:";
    for(int i=1;i<=tr[root].sz;i++)getidx(i);
    cout<<endl;
}

const ll seed=1e18+7;

int main()
{
    //freopen("in.txt","r",stdin);
    //srand(seed);
    scanf("%d %d",&n,&m);
    init();
    //check();
    char op[20];
    for(int i=1;i<=m;i++){
        scanf("%s",op);
        switch(op[0])
        {
            case 'I':Insert();break;
            case 'D':Delete();break;
            case 'R':Reverse();break;
            case 'G':Getsum();break;
            case 'M':
            if(op[2]=='K')Makesame();
            else Getmaxsum();
            break;
            default:break;
        }

        //check();
    }
    return 0;
}
```

3.6.5 treap 记录父亲

```
//维护父亲
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=8e4+5;
const int INF=1e9+9;

int n,m;
int c[maxn];
queue<int>zzk;
int s[maxn],top;

int T[maxn];
struct node
{
    int l,r,rnd,sz,fa;
    int val;
}tr[maxn];
int tot,root;

void gc(int x){
    if(!x)return;
    zzk.push(x);
    gc(tr[x].l);gc(tr[x].r);
}

int newnode(int v)
{
    int x=0;
    if(zzk.empty())x=++tot;
    else x=zzk.front(),zzk.pop();
    tr[x].sz=1;
    tr[x].l=0;tr[x].r=0;
    tr[x].rnd=rand();
    tr[x].val=v;
    tr[x].fa=0;
    return x;
}

void push_up(int x){
    tr[x].sz=tr[tr[x].l].sz+tr[tr[x].r].sz+1;
    tr[tr[x].l].fa=tr[tr[x].r].fa=x;
}

int build(int *a,int n)
{
    int x,y;
    top=0;
    for(int i=1;i<=n;i++){
        x=newnode(a[i]);y=0;T[a[i]]=x;
        while(top && tr[x].rnd<tr[s[top]].rnd){y=s[top--];push_up(y);}
        if(top)tr[s[top]].r=x;
        tr[x].l=y;
        s[++top]=x;
    }
    while(top)push_up(s[top--]);
    return s[1];
}

void init(){
    tot=0;root=0;
}

int merge(int x,int y)//x<y
{
    if(!x || !y)return x+y;
    if(tr[x].rnd<tr[y].rnd)
    {
        tr[x].r=merge(tr[x].r,y);
        push_up(x);
        return x;
    }
    else
    {
        tr[y].l=merge(x,tr[y].l);
        push_up(y);
        return y;
    }
}

//按size分
void split(int now,int k,int &x,int &y){
    if(!now)x=y=0;
    else
    {
        if(tr[tr[now].l].sz>=k)
        {
            y=now;
            split(tr[now].l,k,x,tr[now].l);
        }
        else
        {
            x=now;
            split(tr[now].r,k-tr[tr[now].l].sz-1,tr[now].r,y);
        }
        push_up(now);
    }
}

int getrank(int now){
    int sum=tr[tr[now].l].sz+1;
    while(tr[now].fa){
        if(tr[tr[now].fa].r==now)sum+=tr[tr[tr[now].fa].l].sz+1;
        now=tr[now].fa;
    }
    return sum;
}

void Top()
{
    int val,rk;
    int x1,y1,x2,y2;
    scanf("%d",&val);
    rk=getrank(T[val]);
    split(root,rk,x1,y1);
    split(x1,rk-1,x2,y2);
    root=merge(y2,merge(x2,y1));
}

void Bottom()
{
    int val,rk;
    int x1,y1,x2,y2;
    scanf("%d",&val);
    rk=getrank(T[val]);
    split(root,rk,x1,y1);
    split(x1,rk-1,x2,y2);
    root=merge(merge(x2,y1),y2);
}

void Insert()
{
    int val,t,rk;
    int x1,y1,x2,y2,x3,y3;
    scanf("%d %d",&val,&t);
    if(t==0)return;
    rk=getrank(T[val]);
    if(t==1){
        split(root,rk+1,x1,y1);
        split(x1,rk-1,x2,y2);
        split(y2,1,x3,y3);
        root=merge(merge(x2,merge(y3,x3)),y1);
    }
    else if(t==-1){
        split(root,rk,x1,y1);
        split(x1,rk-2,x2,y2);
        split(y2,1,x3,y3);
        root=merge(merge(x2,merge(y3,x3)),y1);
    }
}

void Ask()
{
    int k,rk,x,y;
    scanf("%d",&k);
    rk=getrank(T[k]);
    split(root,rk,x,y);
    printf("%d\n",tr[x].sz-1);
    root=merge(x,y);
}

void Query()
{
    int k;
    int x1,y1,x2,y2;
    scanf("%d",&k);
    split(root,k,x1,y1);
    split(x1,k-1,x2,y2);
    printf("%d\n",tr[y2].val);
    root=merge(merge(x2,y2),y1);
}


void getidx(int idx){
    int x,y,z,w;
    split(root,idx-1,x,y);
    split(y,1,z,w);
    int ans=tr[z].val;
    root=merge(x,merge(z,w));
    printf("%d ",ans);
}
void check()
{
    cout<<"CHECK:";
    for(int i=1;i<=tr[root].sz;i++)getidx(i);
    cout<<endl;
}


int main()
{
    srand(time(0));


    scanf("%d %d",&n,&m);

    init();
    for(int i=1;i<=n;i++)scanf("%d",&c[i]);
    root=build(c,n);

    //check();

    char op[20];
    int x,y;
    for(int i=1;i<=m;i++){
        scanf("%s",op);
        switch(op[0])
        {
            case 'T':Top();break;
            case 'B':Bottom();break;
            case 'I':Insert();break;
            case 'A':Ask();break;
            case 'Q':Query();break;
            default:break;
        }
        //check();
    }
    return 0;
}
```

3.6.6 可持久化 treap

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=5e5+5;
const int INF=2147483647;

queue<int>zzk;
```

```
struct node
{
   int l,r,v,rnd,sz;
}tr[maxn<<6];
int tot;

inline void update(int x)
{
   tr[x].sz=tr[tr[x].l].sz+tr[tr[x].r].sz+1;
}

inline int newnode(int v)
{
   int x;
   if(zzk.empty())x=++tot;
   else x=zzk.front(),zzk.pop();
   tr[x].sz=1;tr[x].v=v;tr[x].rnd=rand();
   return x;
}

int merge(int x,int y)//x<y
{
   if(!x || !y)return x+y;
   if(tr[x].rnd<tr[y].rnd)
   {
      tr[x].r=merge(tr[x].r,y);
      update(x);
      return x;
   }
   else
   {
      tr[y].l=merge(x,tr[y].l);
      update(y);
      return y;
   }
}

void split(int now,int k,int &x,int &y){
   if(!now)x=y=0;
   else
   {
      if(tr[now].v<=k)
      {
         x=++tot;tr[x]=tr[now];
         split(tr[x].r,k,tr[x].r,y);
         update(x);
      }
      else
      {
         y=++tot;tr[y]=tr[now];
         split(tr[y].l,k,x,tr[y].l);
         update(y);
      }
   }
}

inline int Getkth(int now,int k)
{
   while(true)
   {
      if(k<=tr[tr[now].l].sz)now=tr[now].l;
      else
      {
         if(k==tr[tr[now].l].sz+1)return tr[now].v;
         else
```

```
            {
                k-=tr[tr[now].l].sz+1;
                now=tr[now].r;
            }
        }
    }
}

inline void Insert(int &rt,int v)
{
    int x,y;
    split(rt,v,x,y);
    rt=merge(merge(x,newnode(v)),y);
}

inline void Delete(int &rt,int v)
{
    int x,y,z;
    split(rt,v,x,z);
    split(x,v-1,x,y);

    y=merge(tr[y].l,tr[y].r);
    rt=merge(merge(x,y),z);
}

inline int Getrank(int &rt,int v)
{
    int x,y;
    split(rt,v-1,x,y);
    int res=tr[x].sz+1;
    rt=merge(x,y);
    return res;
}

inline int Getpre(int &rt,int v)
{
    int x,y;
    split(rt,v-1,x,y);


    if(!x)return -INF;

    int res=x;
    while(tr[res].r)res=tr[res].r;

    rt=merge(x,y);
    return tr[res].v;
}

inline int Getsuc(int &rt,int v)
{
    int x,y;
    split(rt,v,x,y);


    if(!y)return INF;

    int res=y;
    while(tr[res].l)res=tr[res].l;

    rt=merge(x,y);
    return tr[res].v;
}
```

```
void dfs(int x){
    if(tr[x].l)dfs(tr[x].l);
    cout<<tr[x].v<<" ";
    if(tr[x].r)dfs(tr[x].r);
}


int n,m;
int T[maxn];

void init()
{
    tot=0;T[0]=0;
}

int main()
{
    init();
    scanf("%d",&n);

    int ver,op,val;
    for(int i=1;i<=n;i++){
        scanf("%d %d %d",&ver,&op,&val);
        T[i]=T[ver];
        switch(op)
        {
            case 1:Insert(T[i],val);break;
            case 2:Delete(T[i],val);break;
            case 3:printf("%d\n",Getrank(T[i],val));break;
            case 4:printf("%d\n",Getkth(T[i],val));break;
            case 5:printf("%d\n",Getpre(T[i],val));break;
            case 6:printf("%d\n",Getsuc(T[i],val));break;
            default:break;
        }

        //cout<<"AFT "<<T[i]<<":";dfs(T[i]);cout<<endl;
    }
    return 0;
}
```

3.7 LCT

3.7.1 维护树链信息

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=3e5+5;

namespace Link_Cut_Tree{
    #define ls ch[x][0]
    #define rs ch[x][1]
    int ch[maxn][2],fa[maxn],sz[maxn],val[maxn],rev[maxn];
    int sum[maxn];

    inline void clear(int x){ch[x][0]=ch[x][1]=fa[x]=sz[x]=val[x]=rev[x]=0;}

    inline int getch(int x){return (ch[fa[x]][1]==x);}

    inline int isroot(int x){return (ch[fa[x]][0]!=x && ch[fa[x]][1]!=x);}

    inline void reverse(int x){swap(ls,rs);rev[x]^=1;}

    inline void push_up(int x){
        sz[x]=sz[ls]+sz[rs]+1;
        sum[x]=sum[ls]^sum[rs]^val[x];
    }

    inline void push_down(int x){
        if(rev[x]){
            reverse(ls);reverse(rs);
            rev[x]=0;
        }
    }

    void update(int x){
        if(!isroot(x))update(fa[x]);
        push_down(x);
    }

    inline void rotate(int x){
        int f=fa[x],g=fa[f],c=getch(x);
        if(!isroot(f))ch[g][getch(f)]=x;
        fa[x]=g;
        ch[f][c]=ch[x][c^1]; fa[ch[f][c]]=f;
        ch[x][c^1]=f;fa[f]=x;
        push_up(f);push_up(x);
    }

    inline void splay(int x){
        update(x);
        for(;!isroot(x);rotate(x))
        if(!isroot(fa[x]))rotate(getch(fa[x])==getch(x)?fa[x]:x);
    }

    inline void access(int x){
        for(int y=0;x;y=x,x=fa[x])splay(x),rs=y,push_up(x);
    }

    inline void makeroot(int x){
        access(x);splay(x);reverse(x);
    }

    inline int findroot(int x){
        access(x);splay(x);
        while(ls)push_down(x),x=ls;
        return x;
    }

    inline void link(int x,int y){
        makeroot(x);
        if(findroot(y)!=x) fa[x]=y;
    }

    inline void cut(int x,int y)
    {
        makeroot(x);
        if(findroot(y)==x && fa[x]==y && ch[y][0]==x && !ch[y][1]){
            fa[x]=ch[y][0]=0;
            push_up(y);
        }
    }

    inline void split(int x, int y)
    {
        makeroot(x); access(y); splay(y);
    }
}
```

```
using namespace Link_Cut_Tree;

int main()
{
    int n,m;
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++)scanf("%d",&val[i]);
    int op,u,v;
    while(m--){
        scanf("%d%d%d",&op,&u,&v);
        switch(op)
        {
            case 0:split(u,v);printf("%d\n",sum[v]);break;
            case 1:link(u,v);break;
            case 2:cut(u,v);break;
            case 3:val[u]=v;splay(u);break;
        }
    }
    return 0;
}
```

3.7.2 维护边双

```
//并查集缩点
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=3e4+5;
const int maxm=1e5+5;
const int maxq=4e4+5;

namespace Union_Find_Set{
    int f[maxn];
    inline void init(int n){
        for(int i=1;i<=n;i++)f[i]=i;
    }
    int find(int x){return f[x]==x?x:(f[x]=find(f[x]));}
    void Union(int x,int y){
        x=find(x);y=find(y);
        if(x!=y)f[x]=y;
    }
}using namespace Union_Find_Set;

namespace Link_Cut_Tree{
    #define ls ch[x][0]
    #define rs ch[x][1]
    int ch[maxn][2],fa[maxn],sz[maxn],val[maxn],rev[maxn];

    inline void clear(int x){ch[x][0]=ch[x][1]=fa[x]=sz[x]=val[x]=rev[x]=0;}

    inline int getch(int x){return (ch[fa[x]][1]==x);}

    inline int isroot(int x){return (ch[fa[x]][0]!=x && ch[fa[x]][1]!=x);}

    inline void reverse(int x){swap(ls,rs);rev[x]^=1;}

    inline void push_up(int x){
        sz[x]=sz[ls]+sz[rs]+1;
    }

    inline void push_down(int x){
        if(rev[x]){
            reverse(ls);reverse(rs);
            rev[x]=0;
        }
    }

    void update(int x){
        if(!isroot(x))update(fa[x]);
        push_down(x);
    }

    inline void rotate(int x){
        int f=fa[x],g=fa[f],c=getch(x);
        if(!isroot(f))ch[g][getch(f)]=x;
        fa[x]=g;
        ch[f][c]=ch[x][c^1]; fa[ch[f][c]]=f;
        ch[x][c^1]=f;fa[f]=x;
        push_up(f);push_up(x);
    }

    inline void splay(int x){
        update(x);
        for(;!isroot(x);rotate(x))
        if(!isroot(fa[x]))rotate(getch(fa[x])==getch(x)?fa[x]:x);
    }

    inline void access(int x){
        for(int y=0;x;y=x,x=fa[y]=find(fa[x]))splay(x),rs=y,push_up(x);
    }

    inline void makeroot(int x){
        access(x);splay(x);reverse(x);
    }

    inline int findroot(int x){
        access(x);splay(x);
        while(ls)push_down(x),x=ls;
        splay(x);
        return x;
    }

    inline void link(int x,int y){
        makeroot(x);
        if(findroot(y)!=x) fa[x]=y;
    }

    inline void cut(int x,int y)
    {
        makeroot(x);
        if(findroot(y)==x && fa[x]==y && ch[y][0]==x && !ch[y][1]){
            fa[x]=ch[y][0]=0;
            push_up(y);
        }
    }

    inline void split(int x, int y)
    {
        makeroot(x); access(y); splay(y);
    }

    void dfs(int x,int y){
        push_down(x);
        f[x]=y;
        if(ch[x][0]) dfs(ch[x][0],y);
        if(ch[x][1]) dfs(ch[x][1],y);
    }

    inline void merge(int x,int y){
```

```cpp
        if(x==y)return;
        makeroot(x);
        if(findroot(y)!=x)
        fa[x]=y;
        else{
            dfs(rs,x);
            rs=0; push_up(x);
        }
    }
}using namespace Link_Cut_Tree;

struct Edge{
    int u,v;
    bool operator<(const Edge& x)const{if(u==x.u)return v<x.v;return u<x.u;}
}e[maxm];
bool vis[maxm];

struct Query{
    int op,u,v;
}q[maxq];
int ans[maxq],Q;

int n,m;
int main()
{
    scanf("%d %d",&n,&m);
    init(n);
    int u,v;
    for(int i=1;i<=m;i++){
        scanf("%d %d",&u,&v);
        if(u>v)swap(u,v);
        e[i]=(Edge){u,v};
    }
    sort(e+1,e+1+m);
    int op;
    while(scanf("%d",&op) && op!=-1){
        scanf("%d %d",&u,&v);
        if(!op){
            if(u>v)swap(u,v);
            vis[lower_bound(e+1,e+1+m,(Edge){u,v})-e]=1;
        }
        q[++Q]=(Query){op,u,v};
    }

    for(int i=1;i<=m;i++){
        if(!vis[i]){
            u=find(e[i].u); v=find(e[i].v);
            merge(u,v);
        }
    }

    int id=0;
    for(int i=Q;i>=1;i--){
        u=find(q[i].u); v=find(q[i].v);
        if(q[i].op)
        split(u,v), ans[++id]=sz[v]-1;
        else
        merge(u,v);
    }
    for(int i=id;i>=1;i--)printf("%d\n",ans[i]);
    return 0;
}
```

3.7.3 维护 MST 删边

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=1e3+5;
const int maxm=1e5+5;
const int maxq=1e5+5;
const int N=maxn+maxm;
const int INF=1e9+7;

int n,m,Q;
struct Edge{
    int u,v,w;
    bool operator<(const Edge& x){return w<x.w;}
}e[maxm];
bool vis[maxm];
int ans[maxq],pos[maxn][maxn];
struct Query{
    int op,u,v;
}q[maxq];

namespace Link_Cut_Tree{
    #define ls ch[x][0]
    #define rs ch[x][1]
    int ch[N][2],fa[N],sz[N],val[N],rev[N];
    int ma[N];

    inline void clear(int x){ch[x][0]=ch[x][1]=fa[x]=sz[x]=val[x]=rev[x]=ma[x]=0;}

    inline int getch(int x){return (ch[fa[x]][1]==x);}

    inline int isroot(int x){return (ch[fa[x]][0]!=x && ch[fa[x]][1]!=x);}

    inline void reverse(int x){swap(ls,rs);rev[x]^=1;}

    inline void push_up(int x){
        sz[x]=sz[ls]+sz[rs]+1;
        ma[x]=x;
        if(ls && val[ma[ls]]>val[ma[x]])ma[x]=ma[ls];
        if(rs && val[ma[rs]]>val[ma[x]])ma[x]=ma[rs];
    }

    inline void push_down(int x){
        if(rev[x]){
            reverse(ls);reverse(rs);
            rev[x]=0;
        }
    }

    void update(int x){
        if(!isroot(x))update(fa[x]);
        push_down(x);
    }

    inline void rotate(int x){
        int f=fa[x],g=fa[f],c=getch(x);
        if(!isroot(f))ch[g][getch(f)]=x;
        fa[x]=g;
        ch[f][c]=ch[x][c^1]; fa[ch[f][c]]=f;
        ch[x][c^1]=f;fa[f]=x;
        push_up(f);push_up(x);
    }

    inline void splay(int x){
        update(x);
        for(;!isroot(x);rotate(x))
        if(!isroot(fa[x]))rotate(getch(fa[x])==getch(x)?fa[x]:x);
    }

    inline void access(int x){
        for(int y=0;x;y=x,x=fa[x])splay(x),rs=y,push_up(x);
    }

    inline void makeroot(int x){
        access(x);splay(x);reverse(x);
    }

    inline int findroot(int x){
        access(x);splay(x);
        while(ls)push_down(x),x=ls;
        return x;
    }

    inline void link(int x,int y){
        makeroot(x);
        if(findroot(y)!=x) fa[x]=y;
    }

    inline void cut(int x,int y)
    {
        makeroot(x);
        if(findroot(y)==x && fa[x]==y && ch[y][0]==x && !ch[y][1]){
            fa[x]=ch[y][0]=0;
            push_up(y);
        }
    }

    inline void split(int x, int y)
    {
        makeroot(x); access(y); splay(y);
    }

    inline int getmax(int x,int y)
    {
        split(x,y);
        return ma[y];
    }
}using namespace Link_Cut_Tree;

int main()
{
    scanf("%d %d %d",&n,&m,&Q);
    for(int i=1;i<=m;i++){
        scanf("%d %d %d",&e[i].u,&e[i].v,&e[i].w);
        if(e[i].u>e[i].v)swap(e[i].u,e[i].v);
    }
    sort(e+1,e+1+m);
    for(int i=1;i<=m;i++)pos[e[i].u][e[i].v]=i;
    for(int i=1;i<=Q;i++){
        scanf("%d %d %d",&q[i].op,&q[i].u,&q[i].v);
        if(q[i].u>q[i].v)swap(q[i].u,q[i].v);
        if(q[i].op==2){
            vis[pos[q[i].u][q[i].v]]=true;
        }
    }
    for(int i=1;i<=m;i++)val[i+n]=e[i].w;

    int u,v,w,id=0;
    for(int i=1;i<=m;i++){
        if(!vis[i]){
            u=e[i].u; v=e[i].v; w=e[i].w;
            makeroot(u);
            if(findroot(v)==u){
                int ma=getmax(u,v);
                if(val[ma]>w){
                    cut(e[ma-n].u,ma);cut(e[ma-n].v,ma);
                    link(u,i+n);link(v,i+n);
                }
                else continue;
            }
            else{
                link(u,i+n); link(v,i+n);
            }
        }
    }
    for(int i=Q;i>=1;i--){
        u=q[i].u; v=q[i].v;
        if(q[i].op==1){
            split(u,v);
            ans[++id]=val[ma[v]];
        }
        else{
            int p=pos[u][v]; w=e[p].w;
            makeroot(u);
            if(findroot(v)==u){
                int ma=getmax(u,v);
                if(val[ma]>w){
                    cut(e[ma-n].u,ma);cut(e[ma-n].v,ma);
                    link(u,p+n);link(v,p+n);
                }
                else continue;
            }
            else{
                link(u,p+n); link(v,p+n);
            }
        }
    }
    for(int i=id;i>=1;i--)printf("%d\n",ans[i]);
    return 0;
}
```

3.7.4 维护 MST 改边权

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=2e4+5;
const int maxm=5e4+5;
const int maxq=5e4+5;
#define N maxm<<2

struct Edge{
    int u,v,w;
}e[N];

int n,m,q;
int Begin[maxm],pos[maxm];
ll ans[maxq];

namespace Link_Cut_Tree{
    #define ls ch[x][0]
    #define rs ch[x][1]
    int ch[N][2],fa[N],sz[N],val[N],rev[N];
    int ma[N];
```

```
inline void clear(int x){ch[x][0]=ch[x][1]=fa[x]=sz[x]=val[x]=rev[x]=ma[x]=0;}

inline int getch(int x){return (ch[fa[x]][1]==x);}

inline int isroot(int x){return (ch[fa[x]][0]!=x && ch[fa[x]][1]!=x);}

inline void reverse(int x){swap(ls,rs);rev[x]^=1;}

inline void push_up(int x){
    sz[x]=sz[ls]+sz[rs]+1;
    ma[x]=x;
    if(ls && val[ma[ls]]>val[ma[x]])ma[x]=ma[ls];
    if(rs && val[ma[rs]]>val[ma[x]])ma[x]=ma[rs];
}

inline void push_down(int x){
    if(rev[x]){
        reverse(ls);reverse(rs);
        rev[x]=0;
    }
}

void update(int x){
    if(!isroot(x))update(fa[x]);
    push_down(x);
}

inline void rotate(int x){
    int f=fa[x],g=fa[f],c=getch(x);
    if(!isroot(f))ch[g][getch(f)]=x;
    fa[x]=g;
    ch[f][c]=ch[x][c^1]; fa[ch[f][c]]=f;
    ch[x][c^1]=f;fa[f]=x;
    push_up(f);push_up(x);
}

inline void splay(int x){
    update(x);
    for(;!isroot(x);rotate(x))
    if(!isroot(fa[x]))rotate(getch(fa[x])==getch(x)?fa[x]:x);
}

inline void access(int x){
    for(int y=0;x;y=x,x=fa[x])splay(x),rs=y,push_up(x);
}

inline void makeroot(int x){
    access(x);splay(x);reverse(x);
}

inline int findroot(int x){
    access(x);splay(x);
    while(ls)push_down(x),x=ls;
    return x;
}

inline void link(int x,int y){
    makeroot(x);
    if(findroot(y)!=x) fa[x]=y;
}

inline void cut(int x,int y)
{
    makeroot(x);
```

```
      if(findroot(y)==x && fa[x]==y && ch[y][0]==x && !ch[y][1]){
         fa[x]=ch[y][0]=0;
         push_up(y);
      }
   }

   inline void split(int x, int y)
   {
      makeroot(x); access(y); splay(y);
   }

   inline int getmax(int x,int y){
      split(x,y);
      return ma[y];
   }
}using namespace Link_Cut_Tree;

namespace Segment_Tree{//Segment Tree for Segment Tree divide and conquer
   int s[N],st[N],top=0;
   ll Ans=0;
   vector<int>op[N];
   void update(int x,int l,int r,int L,int R,int p){
      if(l>r || L>R)return;
      if(l==L && r==R){op[x].push_back(p);return;}
      int mid=(l+r)>>1;
      if(R<=mid)update(x<<1,l,mid,L,R,p);
      else if(L>mid)update(x<<1|1,mid+1,r,L,R,p);
      else update(x<<1,l,mid,L,mid,p), update(x<<1|1,mid+1,r,mid+1,R,p);
   }
   void solve(int x,int l,int r){
      int now=top,u,v,w;
      for(int i:op[x]){
         u=e[i].u; v=e[i].v; w=e[i].w;
         if(findroot(u)==findroot(v)){
            int mx=getmax(u,v);
            if(val[mx]>w){
               cut(e[mx-n].u,mx); cut(e[mx-n].v,mx); Ans-=e[mx-n].w;
               s[++top]=mx-n; st[top]=1;
            }
            else continue;
         }
         link(i+n,u); link(i+n,v); Ans+=w;
         s[++top]=i; st[top]=0;
      }
      if(l==r)ans[l]=Ans;
      else{
         int mid=(l+r)>>1;
         solve(x<<1,l,mid); solve(x<<1|1,mid+1,r);
      }

      while(top>now){
         if(st[top]){
            link(e[s[top]].u,n+s[top]); link(e[s[top]].v,n+s[top]);
            Ans+=e[s[top]].w;
         }
         else{
            cut(s[top]+n,e[s[top]].u); cut(s[top]+n,e[s[top]].v);
            Ans-=e[s[top]].w;
         }
         --top;
      }
   }
}using namespace Segment_Tree;

int main()
```

```
{
    scanf("%d %d %d",&n,&m,&q);
    int u,v,w,k,d;
    for(int i=1;i<=m;i++){
        scanf("%d %d %d",&u,&v,&w);
        e[i]=(Edge){u,v,w};
        pos[i]=i; Begin[i]=1;
    }

    for(int i=1;i<=q;i++){
        scanf("%d %d",&k,&d);
        update(1,1,q,Begin[k],i-1,pos[k]); Begin[k]=i;
        e[m+i]=e[k]; e[m+i].w=d; pos[k]=m+i;
    }
    for(int i=1;i<=m+q;i++)val[i+n]=e[i].w; //把边看成点
    for(int i=1;i<=m;i++)update(1,1,q,Begin[i],q,pos[i]);
    solve(1,1,q);
    for(int i=1;i<=q;i++)printf("%lld\n",ans[i]);
    return 0;
}
```

3.7.5 维护子树信息

信息需要满足可减，如子树大小, 否则用 Top Tree

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e5+5;
const int N=maxn;
typedef long long ll;
int n,q;

namespace Link_Cut_Tree{
    #define ls ch[x][0]
    #define rs ch[x][1]
    int ch[N][2],fa[N],sz[N],val[N],rev[N];
    int sz2[N];

    inline void clear(int x)
    {
        ch[x][0]=ch[x][1]=fa[x]=sz[x]=val[x]=rev[x]=0;
    }

    inline int getch(int x)
    {
        return (ch[fa[x]][1]==x);
    }

    inline int isroot(int x)
    {
        return (ch[fa[x]][0]!=x && ch[fa[x]][1]!=x);
    }

    inline void reverse(int x)
    {
        swap(ls,rs);rev[x]^=1;
    }

    inline void push_up(int x)
    {
        sz[x]=sz[ls]+sz[rs]+1+sz2[x];
    }

    inline void push_down(int x)
    {
        if(rev[x]){
```

```
        reverse(ls);reverse(rs);
        rev[x]=0;
    }
}

inline void update(int x)
{
    if(!isroot(x))update(fa[x]);
    push_down(x);
}

inline void rotate(int x)
{
    int f=fa[x],g=fa[f],c=getch(x);
    if(!isroot(f))ch[g][getch(f)]=x;
    fa[x]=g;
    ch[f][c]=ch[x][c^1]; fa[ch[f][c]]=f;
    ch[x][c^1]=f;fa[f]=x;
    push_up(f);push_up(x);
}

inline void splay(int x)
{
    update(x);
    for(;!isroot(x);rotate(x)) if(!isroot(fa[x]))rotate(getch(fa[x])==getch(x)?fa[x]:x);
}

inline void access(int x)
{
    for(int y=0;x;y=x,x=fa[x])
    splay(x),sz2[x]+=sz[ch[x][1]]-sz[y], rs=y, push_up(x); //attention
}

inline void makeroot(int x)
{
    access(x);splay(x);reverse(x);
}

inline int findroot(int x)
{
    access(x);splay(x);
    while(ls)push_down(x),x=ls;
    return x;
}

inline void link(int x,int y)
{
    makeroot(x);
    if(findroot(y)!=x)fa[x]=y, sz2[y]+=sz[x]; //attention
}

inline void cut(int x,int y)
{
    makeroot(x);
    if(findroot(y)==x && fa[x]==y && ch[y][0]==x && !ch[y][1])
    fa[x]=ch[y][0]=0, push_up(y);
}

inline void split(int x,int y)
{
    makeroot(x); access(y); splay(y);
}

inline ll getans(int x,int y)
{
        cut(x,y);
        splay(x); splay(y);
        ll ans=1LL*sz[x]*sz[y];
        link(x,y);
        return ans;
    }
}using namespace Link_Cut_Tree;

int main()
{
    int n,q;
    scanf("%d %d",&n,&q);
    char op[5];
    int x,y;
    while(q--){
        scanf("%s%d%d",op,&x,&y);
        if(op[0]=='A'){
            link(x,y);
        }
        else{
            printf("%lld\n",getans(x,y));
        }
    }
    return 0;
}
```

3.7.6 LCA

1. 将当前查询需要的根 makeroot, 表示为当前根下的 lca
2. 将 x Access 到根, 此时 x 到根的路径是一棵平衡树
3. 将 y Access, 中途遇到了与根一棵平衡树的点就记录下来 (即与 x 到根的路径相遇), 返回即可

```
int access(int x){
    int tmp = 0;
    for(int y=0;x;y=x,x=fa[x])
        splay(x), rs = y, tmp = x;
    return tmp;
}
int LCA(int root,int x,int y) {
    makeroot(root);
    access(x); return access(y);
}
```

3.7.7 树的重心

```
/*
1.连接两棵树, 新的重心在两个原重心的路径上, 在这条路径上寻找答案
2.具体找法: 类似树上二分, 我们需要不断逼近树的重心的位置。记下lsum表示当前链中搜索区间左端点以左的子树大小, rsum表示右端点
    以右的。x的整个子树就表示了当前搜索区间, 在中序遍历中x把搜索区间分成了左右两块 (在Splay中对应x的左子树和右子树) 。
3.如果x左子树的s加上lsum和x右子树的s加上rsum都不超过新树总大小的一半, 那么x当然就是重心啦! 当然, 如果总大小是奇数, 重心只
    会有一个, 那就找到了。否则, 因为必须编号最小, 所以还要继续找下去。
4.当我们没有确定答案时, 还要继续找下去, 那么就要跳儿子了。x把整个链分成了左右两个部分, 而重心显然会在大小更大的一部分中, 这
    个也应该好证明。如果x左子树的s加上lsum小于x右子树的s加上rsum, 那就跳右儿子继续找。这时候当前搜索区间减小了, 搜索区间以
    外的部分增大了, lsum应该加上si[x]+1。反之亦然。如果跳进了空儿子, 那肯定所有情况都考虑完了, 直接结束查找。
*/
#include<bits/stdc++.h>
#define R register int
#define I inline void
const int N=100009,INF=2147483647;
int f[N],c[N][2],si[N],s[N],h[N];
bool r[N];
#define lc c[x][0]
#define rc c[x][1]
inline bool nroot(R x){return c[f[x]][0]==x||c[f[x]][1]==x;}
```

```
I pushup(R x){
   s[x]=s[lc]+s[rc]+si[x]+1;
}
I pushdown(R x){
   if(r[x]){
      R t=lc;lc=rc;rc=t;
      r[lc]^=1;r[rc]^=1;r[x]=0;
   }
}
I pushall(R x){
   if(nroot(x))pushall(f[x]);
   pushdown(x);
}
I rotate(R x){
   R y=f[x],z=f[y],k=c[y][1]==x,w=c[x][!k];
   if(nroot(y))c[z][c[z][1]==y]=x;
   f[f[f[c[c[x][!k]=y][k]=w]=y]=x]=z;pushup(y);//为三行rotate打call
}
I splay(R x){
   pushall(x);
   R y;
   while(nroot(x)){
      if(nroot(y=f[x]))rotate((c[f[y]][0]==y)^(c[y][0]==x)?x:y);
      rotate(x);
   }
   pushup(x);
}
I access(R x){
   for(R y=0;x;x=f[y=x]){
      splay(x);
      si[x]+=s[rc];
      si[x]-=s[rc=y];
      pushup(x);
   }
}
I makeroot(R x){
   access(x);splay(x);
   r[x]^=1;
}
I split(R x,R y){
   makeroot(x);
   access(y);splay(y);
}
I link(R x,R y){
   split(x,y);
   si[f[x]=y]+=s[x];
   pushup(y);
}
int geth(R x){
   if(h[x]==x)return x;
   return h[x]=geth(h[x]);
}
inline int update(R x){
   R l,r,ji=s[x]&1,sum=s[x]>>1,lsum=0,rsum=0,newp=INF,nowl,nowr;
   while(x){
      pushdown(x);//注意pushdown
      nowl=s[l=lc]+lsum;nowr=s[r=rc]+rsum;
      if(nowl<=sum&&nowr<=sum){
         if(ji){newp=x;break;}//剪枝，确定已经直接找到
         else if(newp>x)newp=x;//选编号最小的
      }
      if(nowl<nowr)lsum+=s[l]+si[x]+1,x=r;
      else rsum+=s[r]+si[x]+1,x=l;//缩小搜索区间
   }
   splay(newp);//保证复杂度
```

```
    return newp;
}
#define G ch=getchar()
#define gc G;while(ch<'-')G
#define in(z) gc;z=ch&15;G;while(ch>'-')z*=10,z+=ch&15,G;
int main(){
    register char ch;
    R n,m,x,y,z,Xor=0;
    in(n);in(m);
    for(R i=1;i<=n;++i)s[i]=1,h[i]=i,Xor^=i;
    while(m--){
        gc;
        switch(ch){
            case 'A':in(x);in(y);link(x,y);
            split(x=geth(x),y=geth(y));//提出原重心路径
            z=update(y);
            Xor=Xor^x^y^z;
            h[x]=h[y]=h[z]=z;//并查集维护好
            break;
            case 'Q':in(x);printf("%d\n",geth(x));break;
            case 'X':gc;gc;printf("%d\n",Xor);
        }
    }
    return 0;
}
```

3.8 Top Tree

```
#pragma GCC optimize(2)
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=100005, N = maxn << 1, inf = 0x3f3f3f3f, unused = -inf;
int n, m;
namespace Top_Tree{
    /*
    * rooted tree
    * *C:Chain infomation
    * *S:Virtual Subtree infomation
    * *T:C+S
    */
    #define l ch[x][0]
    #define r ch[x][1]
    #define ls ch[x][2]
    #define rs ch[x][3]
    int fa[N], ch[N][4], val[N], in[N], addC[N], tagC[N], addS[N], tagS[N], stk[N], rev[N], tot, edge[N][2];
    queue<int> que;
    struct data {
        int sum, minVal, maxVal, size;
        data () {sum = size = 0, minVal = inf, maxVal = -inf;}
        data (int x) {sum = minVal = maxVal = x, size = 1;}
        data (int x, int y, int z, int w) : sum(x), minVal(y), maxVal(z), size(w) {}
        inline data operator + (const data &a) const {
            return data(sum + a.sum, min(minVal, a.minVal), max(maxVal, a.maxVal), size + a.size);
        }
    }sumC[N], sumS[N], sumT[N];

    inline int getch(int x) {
        for (int i = 1; i <= 3; ++i) if (ch[fa[x]][i] == x)
        return i;
        return 0;
    }

    inline int isRoot(int x, int t) {
        if (t) return !fa[x] || !in[fa[x]] || !in[x];
        return !fa[x] || (ch[fa[x]][0] != x && ch[fa[x]][1] != x) || in[fa[x]] || in[x];
    }

    inline void pushup(int x) {
        if (!x) return;
        sumC[x] = in[x] ? data() : data(val[x]), sumS[x] = data();
        if (l) sumC[x] = sumC[x] + sumC[l], sumS[x] = sumS[x] + sumS[l];
        if (r) sumC[x] = sumC[x] + sumC[r], sumS[x] = sumS[x] + sumS[r];
        if (ls) sumS[x] = sumS[x] + sumT[ls];
        if (rs) sumS[x] = sumS[x] + sumT[rs];
        sumT[x] = sumC[x] + sumS[x];
    }

    inline void rotate(int x, int t) {
        int f = fa[x], gf = fa[f], k = getch(x);
        ch[f][k] = ch[x][k ^ 1], fa[ch[x][k ^ 1]] = f;
        if (gf) ch[gf][getch(f)] = x;
        fa[f] = x, fa[x] = gf, ch[x][k ^ 1] = f;
        pushup(f), pushup(x);
    }

    inline void modifyAddC(int x, int w) {
        sumC[x].maxVal += w, sumC[x].minVal += w, sumC[x].sum += w * sumC[x].size;
        val[x] += w, addC[x] += w, sumT[x] = sumC[x] + sumS[x];
    }

    inline void modifyTagC(int x, int w) {
        if (sumC[x].size)
        sumC[x].maxVal = sumC[x].minVal = w;
        sumC[x].sum = w * sumC[x].size;
        val[x] = w, tagC[x] = w, addC[x] = 0, sumT[x] = sumC[x] + sumS[x];
    }

    inline void modifyAddS(int x, int w, int t) {
        sumS[x].maxVal += w, sumS[x].minVal += w, sumS[x].sum += w * sumS[x].size, addS[x] += w;
        (t && !in[x]) ? modifyAddC(x, w) : void (sumT[x] = sumC[x] + sumS[x]);
    }

    inline void modifyTagS(int x, int w, int t) {
        if (sumS[x].size)
        sumS[x].maxVal = w, sumS[x].minVal = w;
        sumS[x].sum = w * sumS[x].size, addS[x] = 0, tagS[x] = w;
        (t && !in[x]) ? modifyTagC(x, w) : void (sumT[x] = sumC[x] + sumS[x]);
    }

    inline void modifyRev(int x) {
        rev[x] ^= 1, swap(l, r);
    }

    inline void pushdown(int x) {
        if (rev[x]) {
            if (l) modifyRev(l);
            if (r) modifyRev(r);
            rev[x] = 0;
        }
        if (tagC[x] != unused) {
            if (l) modifyTagC(l, tagC[x]);
            if (r) modifyTagC(r, tagC[x]);
            tagC[x] = unused;
        }
        if (tagS[x] != unused) {
            if (l) modifyTagS(l, tagS[x], 0);
            if (r) modifyTagS(r, tagS[x], 0);
            if (ls) modifyTagS(ls, tagS[x], 1);
            if (rs) modifyTagS(rs, tagS[x], 1);
            tagS[x] = unused;
        }
        if (addC[x]) {
            if (l) modifyAddC(l, addC[x]);
            if (r) modifyAddC(r, addC[x]);
            addC[x] = 0;
        }
        if (addS[x]) {
            if (l) modifyAddS(l, addS[x], 0);
            if (r) modifyAddS(r, addS[x], 0);
            if (ls) modifyAddS(ls, addS[x], 1);
            if (rs) modifyAddS(rs, addS[x], 1);
            addS[x] = 0;
        }
    }

    inline int newNode() {
        int ret = que.empty() ? ++tot : que.front();
        if (!que.empty()) que.pop();
        tagS[ret] = unused, addS[ret] = 0, in[ret] = 1;
        return ret;
    }

    inline void splay(int x, int t) {
        if (!x) return;
        int fx = x, ind = 0;
        while (!isRoot(fx, t))
        stk[++ind] = fx, fx = fa[fx];
        pushdown(fx);
        while (ind) pushdown(stk[ind--]);
        while (!isRoot(x, t))
        (!isRoot(fa[x], t) && getch(x) == getch(fa[x])) ? rotate(fa[x], t) : rotate(x, t);
    }

    inline void insrt(int x, int y) {
        pushdown(x);
        if (!ls || !rs) {
            fa[y] = x, ch[x][ls ? 3 : 2] = y;
            return pushup(x);
        }
        int now = x, f = newNode();
        while (ch[now][2] && in[ch[now][2]])
        pushdown(now), now = ch[now][2];
        pushdown(now);
        fa[ch[now][2]] = f, fa[y] = f, fa[f] = now;
        ch[f][2] = ch[now][2], ch[f][3] = y, ch[now][2] = f;
        pushup(f), pushup(now), splay(f, 2), pushup(x);
    }

    inline void delet(int x) {
        splay(x, 0);
        int f = fa[x], gf = fa[f], k = getch(x), tp = fa[x], ind = 0;
        if (!f || !in[f])
        return fa[x] = ch[f][k] = 0, pushup(f), splay(f, 0);
        while (!isRoot(tp, 2)) stk[++ind] = tp, tp = fa[tp];
        fa[x] = 0, pushdown(tp);
        while (ind) pushdown(stk[ind--]);
        if (gf) ch[gf][getch(f)] = ch[f][k ^ 1], fa[ch[f][k ^ 1]] = gf;
        que.push(f), pushup(gf), splay(gf, 2), pushup(fa[gf]);
    }

    inline int anc(int x) {
        int f = fa[x];
        if (!f || !in[f]) return f;
        return splay(f, 2), fa[f];
    }

    inline void access(int x) {
        for (int i = 0; x; i = x, x = anc(x)) {
            splay(x, 0);
            if (i) delet(i);
            if (r) insrt(x, r);
            r = i, fa[i] = x, pushup(x);
        }
    }

    inline void makeRoot(int x) {
        access(x), splay(x, 0), modifyRev(x);
    }

    inline void link(int x, int y) {
        makeRoot(x), insrt(y, x), access(x);
    }

    inline void cut(int x) {
        access(x), splay(x, 0);
        l = fa[l] = 0;
        pushup(x);
    }
    //cut(int x,int y):同LCT

    inline void split(int x, int y) {
        makeRoot(x), access(y), splay(x, 0);
    }

    inline int queryTree(int root, int x, int op) {
        makeRoot(root), access(x), splay(x, 0);
        int ret = val[x];
        if (op == 1) {
            if (ls) ret += sumT[ls].sum;
            if (rs) ret += sumT[rs].sum;
        }
        else if (op == 2) {
            if (ls) ret = max(ret, sumT[ls].maxVal);
            if (rs) ret = max(ret, sumT[rs].maxVal);
        }
        else {
            if (ls) ret = min(ret, sumT[ls].minVal);
            if (rs) ret = min(ret, sumT[rs].minVal);
        }
        return ret;
    }
}using namespace Top_Tree;

int main() {
    int op, x, y, w, root;
    scanf("%d %d",&n,&m); tot = n;
    for (int i = 1; i <= n; ++i)tagC[i] = tagS[i] = unused;
    for (int i = 1; i < n; ++i)scanf("%d %d",&edge[i][0],&edge[i][1]);
    for (int i = 1; i <= n; ++i)scanf("%d",&val[i]), pushup(i);
    for (int i = 1; i < n; ++i)link(edge[i][0], edge[i][1]);
    scanf("%d",&root);
    while (m--) {
        scanf("%d %d",&op,&x);
        if (op == 0) { //子树置数
            scanf("%d",&y), makeRoot(root), access(x), splay(x, 0);
            if (ls) modifyTagS(ls, y, 1);
            if (rs) modifyTagS(rs, y, 1);
            val[x] = y, pushup(x), splay(x, 0);
        }
        else if (op == 1) //换根（整棵树）
            root = x;
        else if (op == 2) //链置数
            scanf("%d %d",&y,&w), split(x, y), modifyTagC(x, w);
        else if (op == 3) //子树询问min
            printf("%d\n",queryTree(root, x, 3));
        else if (op == 4) //子树询问max
            printf("%d\n",queryTree(root, x, 2));
        else if (op == 5) { //子树加
            scanf("%d",&y), makeRoot(root), access(x), splay(x, 0);
            if (ls) modifyAddS(ls, y, 1);
            if (rs) modifyAddS(rs, y, 1);
            val[x] += y, pushup(x), splay(x, 0);
        }
        else if (op == 6) //链加
            scanf("%d %d",&y,&w), split(x, y), modifyAddC(x, w);
        else if (op == 7) //链询问min
            scanf("%d",&y), split(x, y), printf("%d\n",sumC[x].minVal);
        else if (op == 8) //链询问max
            scanf("%d",&y), split(x, y), printf("%d\n",sumC[x].maxVal);
        else if (op == 9) { //换父亲
            scanf("%d",&y), makeRoot(root), access(y), splay(y, 0), splay(x, 0);
            if (!fa[x]) continue;
            cut(x), link(x, y);
        }
        else if (op == 10) //链询问sum
            scanf("%d",&y), split(x, y), printf("%d\n",sumC[x].sum);
        else if (op == 11) //子树询问sum
            printf("%d\n",queryTree(root, x, 1));
    }
    return 0;
}
```

3.9 Euler Tour Tree

3.9.1 动态树的直径

$$\max_{x,y}\{\mathrm{dis}(x,y)\} = \max_{1\le a\le c\le b\le 2n-1}\{\mathrm{dis}(r,p_a)+\mathrm{dis}(r,p_b)-2\mathrm{dis}(r,p_c)\}. \tag{2}$$

```
//tree.dis[1]:树的直径
//修改边权，询问里u最远的点
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
const int maxn = 1e5 + 100;
vector<vector<pii> >G(maxn);
struct edge{
    int u, v; ll w;
}E[maxn];
ll dis[maxn];
int flag[2 * maxn], Eulerorder[2 * maxn], in[maxn], out[maxn];
int pre[maxn];
int dfsclock;
void dfs(int cur, int fa, int dep){
    int size = G[cur].size();
    pre[cur] = fa;
    flag[++dfsclock] = cur;
    in[cur] = dfsclock;
    Eulerorder[dfsclock] = dep;
```

```
   for (int i = 0; i < size; i++){
      int& v = G[cur][i].first, w = G[cur][i].second;
      if (v == fa)continue;
      dis[v] = dis[cur] + E[w].w;
      dfs(v, cur, dep + 1);
      flag[++dfsclock] = cur;
      Eulerorder[dfsclock] = dep;
   }
   out[cur] = dfsclock;
}
template<typename T>
class Euler_tour_tree{
   public:
   int size;
   vector<T>lazy, disToRoot, preLCA, lpLCA, rpLCA, dis;
   vector<int>node, lpNode, rpNode, lNode, rNode;
   Euler_tour_tree(int _size, int* _flag, T* _dis) :size(_size), flag(_flag), dep(_dis) {
      int sz = size * 4 + 5;
      dis.resize(sz), disToRoot.resize(sz), lazy.resize(sz), lNode.resize(sz), lpLCA.resize(sz), lpNode.resize(
          sz);
      node.resize(sz), preLCA.resize(sz), rNode.resize(sz), rpLCA.resize(sz), rpNode.resize(sz);
      build(1, size);
   }
   void modify(const int l, const int r, const T val, int __l = 1, int __r = -1, int k = 1){
      if (__r == -1)
      __r = size;
      if (__l >= l && __r <= r) {
         lazy[k] += val;
         pushdown(__l, __r, k);
         return;
      }
      pushdown(__l, __r, k);
      int mid = (__l + __r) >> 1;
      if (l <= mid)
      modify(l, r, val, __l, mid, k << 1);
      if (r > mid)
      modify(l, r, val, mid + 1, __r, k << 1 | 1);
      pushdown(__l, mid, k << 1);
      pushdown(mid + 1, __r, k << 1 | 1);
      if (__l != __r)
      update(k);
   }
   T queryLCA(const int l, const int r, int __l = 1, int __r = -1, int k = 1){
      if (__r == -1)
      __r = size;
      if (__l >= l && __r <= r)
      return preLCA[k];
      int mid = (__l + __r) >> 1;
      if (r > mid && l <= mid) {
         T ansl = queryLCA(l, r, __l, mid, k << 1);
         T ansr = queryLCA(l, r, mid + 1, __r, k << 1 | 1);
         return max(ansl, ansr);
      }
      if (l <= mid)
      return queryLCA(l, r, __l, mid, k << 1);
      return queryLCA(l, r, mid + 1, __r, k << 1 | 1);
   }
   T queryDisToRoot(const int t, int __l = 1, int __r = -1, int k = 1){
      if (__r == -1)
      __r = size;
      if (__l == __r) {
         pushdown(__l, __r, k);
         return disToRoot[k];
      }
      pushdown(__l, __r, k);
      int mid = (__l + __r) >> 1;
      if (t > mid)
      return queryDisToRoot(t, mid + 1, __r, k << 1 | 1);
      return queryDisToRoot(t, __l, mid, k << 1);
   }
   private:
   const int* flag; const T* dep;
   inline void pushdown(int l, int r, int k){
      disToRoot[k] += lazy[k];
      preLCA[k] -= lazy[k] * 2;
      lpLCA[k] -= lazy[k];
      rpLCA[k] -= lazy[k];
      if (l != r)
      lazy[k << 1] += lazy[k], lazy[k << 1 | 1] += lazy[k];
      lazy[k] = 0;
   }
   inline void update(int k){
      if (disToRoot[k << 1] > disToRoot[k << 1 | 1])
      disToRoot[k] = disToRoot[k << 1], node[k] = node[k << 1];
      else
      disToRoot[k] = disToRoot[k << 1 | 1], node[k] = node[k << 1 | 1];
      preLCA[k] = max(preLCA[k << 1], preLCA[k << 1 | 1]);
      if (lpLCA[k << 1] > lpLCA[k << 1 | 1])
      lpLCA[k] = lpLCA[k << 1], lpNode[k] = lpNode[k << 1];
      else
      lpLCA[k] = lpLCA[k << 1 | 1], lpNode[k] = lpNode[k << 1 | 1];
      if (lpLCA[k] < disToRoot[k << 1] + preLCA[k << 1 | 1])
      lpLCA[k] = disToRoot[k << 1] + preLCA[k << 1 | 1], lpNode[k] = node[k << 1];
      if (rpLCA[k << 1] > rpLCA[k << 1 | 1])
      rpLCA[k] = rpLCA[k << 1], rpNode[k] = rpNode[k << 1];
      else
      rpLCA[k] = rpLCA[k << 1 | 1], rpNode[k] = rpNode[k << 1 | 1];
      if (rpLCA[k] < disToRoot[k << 1 | 1] + preLCA[k << 1])
      rpLCA[k] = disToRoot[k << 1 | 1] + preLCA[k << 1], rpNode[k] = node[k << 1 | 1];
      if (dis[k << 1] > dis[k << 1 | 1])
      dis[k] = dis[k << 1], lNode[k] = lNode[k << 1], rNode[k] = rNode[k << 1];
      else
      dis[k] = dis[k << 1 | 1], lNode[k] = lNode[k << 1 | 1], rNode[k] = rNode[k << 1 | 1];
      if (dis[k] < lpLCA[k << 1] + disToRoot[k << 1 | 1])
      dis[k] = lpLCA[k << 1] + disToRoot[k << 1 | 1], lNode[k] = lpNode[k << 1], rNode[k] = node[k << 1 | 1];
      if (dis[k] < rpLCA[k << 1 | 1] + disToRoot[k << 1])
      dis[k] = rpLCA[k << 1 | 1] + disToRoot[k << 1], lNode[k] = node[k << 1], rNode[k] = rpNode[k << 1 | 1];
   }
   void build(int __l, int __r, int k = 1){
      if (__l == __r) {
         int pos = flag[__l];
         disToRoot[k] = dep[pos];
         preLCA[k] = -2 * dep[pos];
         lpLCA[k] = rpLCA[k] = -dep[pos];
         node[k] = lpNode[k] = rpNode[k] = pos;
         return;
      }
      int mid = (__l + __r) >> 1;
      build(__l, mid, k << 1), build(mid + 1, __r, k << 1 | 1);
      update(k);
   }
};

int main()
{
   int n, m;
   cin >> n;
   int u, v, _w;
   for(int i = 1; i <= n; i++){
      cin >> u >> v >> _w;
```

```
        G[u].push_back(make_pair(v, i));
        G[v].push_back(make_pair(u, i));
        E[i] = edge{ u, v, _w };
    }
    dfs(1, 1, 0);
    Euler_tour_tree<ll>tree(2 * n - 1, flag, dis);
    cin >> m;
    char task;
    int x, y;
    while (m--){
        cin >> task;
        if (task == 'C') {
            cin >> x >> y;
            int& u = E[x].u, & v = E[x].v;
            int node = pre[u] == v ? u : v;
            tree.modify(in[node], out[node], y - E[x].w);
            E[x].w = y;
        }
        else {
            cin >> x;
            int node1 = tree.lNode[1], node2 = tree.rNode[1];
            tree.modify(min(in[node1], in[x]), max(in[node1], in[x]), 0);
            ll dix = tree.queryDisToRoot(in[x]);
            ll ans1 = tree.queryDisToRoot(in[node1]) + tree.queryLCA(min(in[node1], in[x]), max(in[node1], in[x]))
                ;
            tree.modify(min(in[node2], in[x]), max(in[node2], in[x]), 0);
            ll ans2 = tree.queryDisToRoot(in[node2]) + tree.queryLCA(min(in[node2], in[x]), max(in[node2], in[x]))
                ;
            cout << max(ans1, ans2) + dix << endl;
        }
    }
    return 0;
}
```

3.9.2 子树修改

```
//Splay维护欧拉序,换父亲, 区间加, 子树求和
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=1e5+5;
const int N=maxn<<1;
const ll INF=0x3f3f3f3f3f3f3f3f;

int n,m,T[N];
ll w[maxn];
struct Edge{
    int v,nxt;
}e[maxn<<1];
int head[maxn],cnt=1;
void addedge(int u,int v){
    e[cnt].v=v;e[cnt].nxt=head[u];head[u]=cnt++;
    e[cnt].v=u;e[cnt].nxt=head[v];head[v]=cnt++;
}

int dfn[N],high[N],low[N],dfs_clock;
void dfs(int u,int fa){
    high[u]=++dfs_clock; dfn[dfs_clock]=u;
    for(int i=head[u];i;i=e[i].nxt){
        if(e[i].v!=fa)dfs(e[i].v,u);
    }
    low[u]=++dfs_clock; dfn[dfs_clock]=-u;
}

//Splay
```

```
int ch[N][2],f[N],sz1[N],sz2[N],type[N];
ll val[N],addv[N],sum[N];

int tot,root;

inline void init(){tot=root=0;}

inline void clear(int x)
{
    ch[x][0]=ch[x][1]=f[x]=val[x]=0;
    val[x]=addv[x]=sum[x]=sz1[x]=sz2[x]=0;
}

inline bool get(int x){return ch[f[x]][1]==x;}

inline void push_up(int x)
{
    int l=ch[x][0],r=ch[x][1];
    sum[x]=val[x]; sz1[x]=(type[x]==1); sz2[x]=(type[x]==-1);
    if(l)sum[x]+=sum[l], sz1[x]+=sz1[l], sz2[x]+=sz2[l];
    if(r)sum[x]+=sum[r], sz1[x]+=sz1[r], sz2[x]+=sz2[r];
}

inline void addval(int x,ll d){
    addv[x]+=d;
    val[x]+=d*type[x];
    sum[x]+=d*(sz1[x]-sz2[x]);
}

inline void push_down(int x)
{
    if(addv[x]){
        if(ch[x][0])addval(ch[x][0],addv[x]);
        if(ch[x][1])addval(ch[x][1],addv[x]);
        addv[x]=0;
    }
}

void update(int x){
    if(f[x])update(f[x]);
    push_down(x);
}

inline void rotate(int x)
{
    int old=f[x],oldf=f[old],whichx=get(x);
    ch[old][whichx]=ch[x][whichx^1]; f[ch[old][whichx]]=old;
    ch[x][whichx^1]=old; f[old]=x;
    f[x]=oldf;
    if(oldf)ch[oldf][ch[oldf][1]==old]=x;
    push_up(old); push_up(x);
}

inline void splay(int x,int ed)
{
    update(x);
    for(int fa;(fa=f[x])!=ed;rotate(x))
    if(f[fa]!=ed)rotate(((get(x)==get(fa))?fa:x));
    if(ed==0)root=x;
}

int build(int l,int r,int fa)
{
    if(l>r)return 0;
    int mid=(l+r)/2,flag=(dfn[mid]<0)?-1:1;
    int now=++tot; T[mid]=now;
    ch[now][0]=ch[now][1]=0; f[now]=fa; type[now]=flag;
    sum[now]=val[now]=flag*w[flag*dfn[mid]]; addv[now]=0;
    sz1[now]=(flag==1); sz2[now]=(flag==-1);

    ch[now][0]=build(l,mid-1,now);
    ch[now][1]=build(mid+1,r,now);

    push_up(now);
    return now;
}

int getpre(int x){
    x=ch[x][0]; while(ch[x][1])x=ch[x][1];
    return x;
}

int getsuc(int x){
    x=ch[x][1]; while(ch[x][0])x=ch[x][0];
    return x;
}

void split(int &u,int &v){
    splay(u,0); u=getpre(u);
    splay(v,0); v=getsuc(v);
    splay(u,0); splay(v,u);
}

void query(int u){
    int p=T[high[1]],q=T[high[u]];
    split(p,q);
    printf("%lld\n",sum[ch[q][0]]);
}

void modify_parent(int u,int v){
    int p=T[high[u]],q=T[low[u]];
    split(p,q);
    int x=ch[q][0];
    ch[q][0]=0; f[x]=0;
    push_up(q); push_up(p);

    p=T[high[v]]; splay(p,0);
    q=getsuc(p); splay(q,p);
    ch[q][0]=x; f[x]=q;
    push_up(q); push_up(p);
}

void modify_add(int u,ll d){
    int p=T[high[u]],q=T[low[u]];
    split(p,q);
    addval(ch[q][0],d);
}

int main()
{
    scanf("%d",&n);
    for(int i=2,f;i<=n;i++)scanf("%d",&f), addedge(i,f);
    for(int i=1;i<=n;i++)scanf("%lld",&w[i]);
    dfs_clock=1; dfs(1,-1); ++dfs_clock;

    init(); root=build(1,dfs_clock,0);
    scanf("%d",&m);
    char op[5]; int x,y,d,p,q;
    while(m--){
        scanf("%s",op);
        if(op[0]=='Q'){
            scanf("%d",&d);
            query(d);
        }
        else if(op[0]=='C'){
            scanf("%d %d",&x,&y);
            modify_parent(x,y);
        }
        else if(op[0]=='F'){
            scanf("%d %d",&p,&q);
            modify_add(p,q);
        }
    }
    return 0;
}
```

3.10 左偏树

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int maxn = 1e5+5;

int c[maxn];
int n,m;
int ls[maxn],rs[maxn],dis[maxn],fa[maxn];

int Merge(int x,int y)
{
    if(!x || !y) return x+y;
    if(c[x]<c[y]) swap(x,y);
    rs[x]=Merge(rs[x],y);
    fa[rs[x]]=x;
    if(dis[ls[x]]<dis[rs[x]]) swap(ls[x],rs[x]);
    if(!rs[x])
        dis[x]=0;
    else
        dis[x]=dis[rs[x]]+1;
    return x;
}

/*
可持久化
void pushdown(int r)
{
    if(!t[r].tag)return;int w=t[r].tag;
    if(t[r].ls)t[t[r].ls].v+=w,t[t[r].ls].tag+=w;
    if(t[r].rs)t[t[r].rs].v+=w,t[t[r].rs].tag+=w;
    t[r].tag=0;
}
int merge(int x, int y) {
    if (!x || !y) return x + y;
    pushdown(x);pushdown(y);
    if (c[x] < c[y]) swap(x, y);
    int p = ++cnt;
    ls[p] = ls[x];
    c[p] = c[x];
    rs[p] = merge(rs[x], y);
    if (dis[ls[p]] < dis[rs[p]]) swap(ls[p], rs[p]);
    dis[p] = dis[rs[p]] + 1;
    return p;
}
*/
```

```
int getroot(int x)
{
   return fa[x]==x?x:fa[x]=getroot(fa[x]);
}

int top(int x){
   if(!ls[x] && !rs[x])
   {
      c[x]/=2;
      return x;
   }

   int nrt=Merge(ls[x],rs[x]);
   fa[nrt]=nrt;
   fa[x]=x;
   ls[x]=rs[x]=0;
   c[x]/=2;
   int nrt1=Merge(nrt,x);
   fa[x]=fa[nrt]=nrt1;
   return nrt1;
}

int duel(int x,int y){
   int rx=getroot(x),ry=getroot(y);
   if(rx==ry) return -1;

   int rtx=top(rx),rty=top(ry);
   int nrt=Merge(rtx,rty);
   fa[rtx]=fa[rty]=nrt;
   return c[nrt];
}

int main()
{
   while(~scanf("%d",&n)){
      for(int i=1;i<=n;i++){
         scanf("%d",&c[i]);
         fa[i]=i;
         ls[i]=0;
         rs[i]=0;
         dis[i]=0;
      }
      scanf("%d",&m);
      int u,v;
      for(int i=1;i<=m;i++){
         scanf("%d %d",&u,&v);
         printf("%d\n",duel(u,v));
      }
   }
   return 0;
}
```

3.11 树链剖分

```
#include<bits/stdc++.h>
#define mid ((l+r)/2)
using namespace std;

typedef long long ll;
const int maxn=3e4+233;
const int maxq=2e5+5;
const int INF=1e9+7;
int w[maxn];
int n,q;
int tot,head[maxn<<6];
struct Edge{
    int v,nxt;
}e[maxn<<6];
void init(){
    tot=0;
    memset(head,-1,sizeof(head));
}
void addedge(int u,int v){
    e[tot].v=v;e[tot].nxt=head[u];
    head[u]=tot++;

    e[tot].v=u;e[tot].nxt=head[v];
    head[v]=tot++;
}



int sz[maxn],son[maxn],fa[maxn],h[maxn],A[maxn],pos[maxn],top[maxn],cnt;
void dfs1(int u,int f){
    int v;
    sz[u]=1;son[u]=0;fa[u]=f;h[u]=h[f]+1;
    for(int i=head[u];i!=-1;i=e[i].nxt){
        v=e[i].v;
        if(v==f)continue;
        dfs1(v,u);

        //w[v]=e[i].w//边权下推成点权

        sz[u]+=sz[v];
        if(sz[son[u]]<sz[v])son[u]=v;
    }
}

void dfs2(int u,int f,int k){
    int v;
    top[u]=k;
    pos[u]=++cnt;
    A[cnt]=w[u];
    if(son[u])dfs2(son[u],u,k);
    for(int i=head[u];i!=-1;i=e[i].nxt){
        v=e[i].v;
        if(v==f)continue;
        if(v==son[u])continue;
        dfs2(v,u,v);
    }
}



int ma[maxn<<4],sum[maxn<<4];
void push_up(int x){
    ma[x]=max(ma[x<<1],ma[x<<1|1]);
    sum[x]=sum[x<<1]+sum[x<<1|1];
}
void build(int x,int l,int r){
    if(l==r){
        ma[x]=A[l];
        sum[x]=A[l];
        return;
    }
    build(x<<1,l,mid);build(x<<1|1,mid+1,r);
    push_up(x);
```

```
}

int getsum(int x,int l,int r,int L,int R){
    if(L==l && R==r)return sum[x];
    if(R<=mid)return getsum(x<<1,l,mid,L,R);
    else if(L>mid)return getsum(x<<1|1,mid+1,r,L,R);
    else return getsum(x<<1,l,mid,L,mid)+getsum(x<<1|1,mid+1,r,mid+1,R);
}

int qsum(int u,int v){
    int ans=0;
    while(top[u]!=top[v]){
        if(h[top[u]]<h[top[v]])swap(u,v);
        ans+=getsum(1,1,n,pos[top[u]],pos[u]);
        u=fa[top[u]];
    }
    if(h[u]>h[v])swap(u,v);
    ans+=getsum(1,1,n,pos[u],pos[v]);
    return ans;
}

int getmax(int x,int l,int r,int L,int R){
    if(L==l && R==r)return ma[x];
    if(R<=mid)return getmax(x<<1,l,mid,L,R);
    else if(L>mid)return getmax(x<<1|1,mid+1,r,L,R);
    else return max(getmax(x<<1,l,mid,L,mid),getmax(x<<1|1,mid+1,r,mid+1,R));
}

int qmax(int u,int v){
    int ans=-INF;
    while(top[u]!=top[v]){
        if(h[top[u]]<h[top[v]])swap(u,v);
        ans=max(ans,getmax(1,1,n,pos[top[u]],pos[u]));
        u=fa[top[u]];
    }
    if(h[u]>h[v])swap(u,v);
    ans=max(ans,getmax(1,1,n,pos[u],pos[v]));
    //若是边权下推成点权，可能需要改成ans=max(ans,getmax(1,1,n,pos[u]+1,pos[v]));
    return ans;
}

void change(int x,int l,int r,int p,int v){
    if(l==p && r==p){
        ma[x]=v;
        sum[x]=v;
        return;
    }
    if(p<=mid)change(x<<1,l,mid,p,v);
    else if(p>mid)change(x<<1|1,mid+1,r,p,v);
    push_up(x);
}

/*
考虑先后顺序
int QUERY(int u,int v){
    node r1,r2,res;
    while(top[u]!=top[v]){
        if(h[top[u]]<h[top[v]]){
            r2=query(1,1,n,pos[top[v]],pos[v])+r2;
            v=fa[top[v]];
        }
        else{
            r1=query(1,1,n,pos[top[u]],pos[u])+r1;
            u=fa[top[u]];
        }
    }


    if(h[u]<h[v])r2=query(1,1,n,pos[u],pos[v])+r2;
    else r1=query(1,1,n,pos[v],pos[u])+r1;


    swap(r1.cl,r1.cr);res=r1+r2;
    return res.num;
}

*/

int main()
{
    scanf("%d",&n);
    init();
    int u,v;
    for(int i=1;i<n;i++){
        scanf("%d %d",&u,&v);
        addedge(u,v);
    }
    for(int i=1;i<=n;i++)scanf("%d",&w[i]);

    cnt=0;
    dfs1(1,0);
    dfs2(1,0,1);
    build(1,1,n);
    scanf("%d",&q);
    char op[233];
    int ans;
    while(q--){
        scanf("%s%d%d",op,&u,&v);

        if(op[0]=='C')change(1,1,n,pos[u],v);
        else
        {
            if(op[1]=='S')ans=qsum(u,v);
            else ans=qmax(u,v);
            printf("%d\n",ans);
        }
    }
    return 0;
}
```

3.12 长链剖分

```
/*
1.类似dsu on tree的技巧，可以把维护子树中只与深度有关的信息做到O(n)
2.对每个点寻找深度最大的儿子作为重儿子，其余作为轻儿子。由此得到了若干条互不相交的长链。
3.在维护信息的过程中，先O(1)继承重儿子的信息，再暴力合并其余轻儿子的信息。
4.因为每个点仅属于一条长链，且一条长链只会在链顶位置作为轻儿子暴力合并一次，所以时间复杂度线性。
*/
//给你一棵树，定义d_{x,i}表示x子树内和x距离为i的节点数，对每个x求使d_{x,i}最大的i，如有多个输出最小的
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;
int gi(){
    int x=0,w=1;char ch=getchar();
    while ((ch<'0'||ch>'9')&&ch!='-') ch=getchar();
    if (ch=='-') w=0,ch=getchar();
    while (ch>='0'&&ch<='9') x=(x<<3)+(x<<1)+ch-'0',ch=getchar();
    return w?x:-x;
}
```

```
const int N = 1e6+5;
int n,to[N<<1],nxt[N<<1],head[N],cnt;
int len[N],son[N],tmp[N],*f[N],*id=tmp,ans[N];
void link(int u,int v){
   to[++cnt]=v;nxt[cnt]=head[u];head[u]=cnt;
   to[++cnt]=u;nxt[cnt]=head[v];head[v]=cnt;
}
void dfs(int u,int ff){
   for (int e=head[u];e;e=nxt[e])
   if (to[e]!=ff){
      dfs(to[e],u);
      if (len[to[e]]>len[son[u]]) son[u]=to[e];
   }
   len[u]=len[son[u]]+1;
}
void dp(int u,int ff){
   f[u][0]=1;
   if (son[u]) f[son[u]]=f[u]+1,dp(son[u],u),ans[u]=ans[son[u]]+1;
   for (int e=head[u];e;e=nxt[e]){
      int v=to[e];if (v==ff||v==son[u]) continue;
      f[v]=id;id+=len[v];dp(v,u);
      for (int j=1;j<=len[v];++j){
         f[u][j]+=f[v][j-1];
         if ((j<ans[u]&&f[u][j]>=f[u][ans[u]])||(j>ans[u]&&f[u][j]>f[u][ans[u]]))
         ans[u]=j;
      }
   }
   if (f[u][ans[u]]==1) ans[u]=0;
}
int main(){
   n=gi();
   for (int i=1;i<n;++i) link(gi(),gi());
   dfs(1,0);f[1]=id;id+=len[1];
   dp(1,0);
   for (int i=1;i<=n;++i) printf("%d\n",ans[i]);
   return 0;
}
```

3.13 树上启发式合并

```
//
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int n,t,c[MAXN],sz[MAXN],st[MAXN],ed[MAXN],cnt[MAXN],rev[MAXN];
vector<int> G[MAXN];
void dfs(int v,int p)
{
   st[v]=++t;rev[t]=v;
   sz[v]=1;
   for(int i=0;i<(int)G[v].size();i++)
   {
      if(G[v][i]==p) continue;
      dfs(G[v][i],v);
      sz[v]+=sz[G[v][i]];
   }
   ed[v]=t;
   return;
}
```

```
void dfs2(int v,int p,bool keep)
{
    int mx=-1,wson=-1;
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p) continue;
        if(sz[to]>mx) {mx=sz[to]; wson=to;}
    }
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p||to==wson) continue;
        dfs2(to,v,0);
    }
    if(wson!=-1) dfs2(wson,v,1);
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p||to==wson) continue;
        for(int j=st[to];j<=ed[to];j++)
        cnt[c[rev[j]]]++;
    }
    cnt[c[v]]++;
    //answer queries here
    if(!keep)
    {
        for(int j=st[v];j<=ed[v];j++)
        cnt[c[rev[j]]]--;
    }
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d",&c[i]);
    for(int i=0;i<n-1;i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        G[u].push_back(v);G[v].push_back(u);
    }
    dfs(1,0);
}
```

3.14 并查集

3.14.1 普通并查集

```
void init()
{
    for(int i=1;i<=n;i++)
    {
        fa[i]=i;
        sz[i]=1;
    }
}
int find(int x)
{
    return x==fa[x]?x:(fa[x]=fin(fa[x]));
}
void merge(int x,int y)
{
    int fx=find(x),fy=find(y);
    if(fx!=fy)
    {
        fa[fx]=fy;
        sz[fy]+=sz[fx];
    }
}
```

3.14.2 可撤销

```
namespace Union_Find_Set{
    int fa[maxn],dep[maxn],p1[maxn],p2[maxn],top;
    void init(int n){
        top=0;
        for(int i=1;i<=n;i++){
            fa[i]=i;
            dep[i]=1;
        }
    }
    int find(int x){return fa[x]==x?x:find(fa[x]);}
    void merge(int x,int y){
        int fx=find(x),fy=find(y);
        if(fx==fy)return;
        if(dep[fx]<=dep[fy]){
            fa[fx]=fy;
            top++; p1[top]=fx; p2[top]=dep[fy];
            dep[fy]=max(dep[fy],dep[fx]+1);
        }
        else{
            fa[fy]=fx;
            top++; p1[top]=fy; p2[top]=dep[fx];
            dep[fx]=max(dep[fx],dep[fy]+1);
        }
    }
    void del(int t){
        while(top>t){
            dep[fa[p1[top]]]=p2[top];
            fa[p1[top]]=p1[top];
            top--;
        }
    }
}using namespace Union_Find_Set;
```

3.14.3 带权并查集

```
long find(int x)
{
    if(x==fa[x]) return x;
    long fx=fa[x];
    fa[x]=find(fa[x]);
    r[x]=(r[x]+r[fx])%3; //维护关系
    return fa[x];
}
void merge(int d,long x,long y)
{
    long fx=find(x),fy=find(y);
    if(fx!=fy)
    {
        fa[fx]=fy;
        r[fx]=(r[y]-r[x]+d+2)%3; //维护关系
    }
}
```

3.14.4 可持久化

```
#include<bits/stdc++.h>
#define N 301000
using namespace std;
template<typename T>inline void read(T &x)
{
    x=0;
    static int p;p=1;
    static char c;c=getchar();
    while(!isdigit(c)){if(c=='-')p=-1;c=getchar();}
    while(isdigit(c)) {x=(x<<1)+(x<<3)+(c-48);c=getchar();}
    x*=p;
}
int n,m;
int L[N*30],R[N*30],fa[N*30],dep[N*30];
int root[N*30];
namespace Persistant_Union_Set
{
    #define Mid ((l+r)>>1)
    #define lson L[rt],l,Mid
    #define rson R[rt],Mid+1,r
    int cnt;
    void build(int &rt,int l,int r)
    {
        rt=++cnt;
        if(l==r){fa[rt]=l;return ;}
        build(lson);build(rson);
    }
    void merge(int last,int &rt,int l,int r,int pos,int Fa)
    {
        rt=++cnt;L[rt]=L[last],R[rt]=R[last];
        if(l==r)
        {
            fa[rt]=Fa;
            dep[rt]=dep[last];
            return ;
        }
        if(pos<=Mid)merge(L[last],lson,pos,Fa);
        else merge(R[last],rson,pos,Fa);
    }
    void update(int rt,int l,int r,int pos)
    {
        if(l==r){dep[rt]++;return ;}
        if(pos<=Mid)update(lson,pos);
        else update(rson,pos);
    }
    int query(int rt,int l,int r,int pos)
    {
        if(l==r)return rt;
        if(pos<=Mid)return query(lson,pos);
        else return query(rson,pos);
    }
    int find(int rt,int pos)
    {
        int now=query(rt,1,n,pos);
        if(fa[now]==pos)return now;
        return find(rt,fa[now]);
    }
    #undef Mid
    #undef lson
    #undef rson
}
using namespace Persistant_Union_Set;
int main()
{
    read(n);read(m);
    build(root[0],1,n);
    for(int i=1;i<=m;i++)
    {
        static int opt,x,y;
        read(opt);read(x);
        if(opt==1)
        {
            read(y);
            static int posx,posy;
            root[i]=root[i-1];
            posx=find(root[i],x);posy=find(root[i],y);
            if(fa[posx]!=fa[posy])
            {
                if(dep[posx]>dep[posy])swap(posx,posy);
                merge(root[i-1],root[i],1,n,fa[posx],fa[posy]);
                if(dep[posx]==dep[posy])update(root[i],1,n,fa[posy]);
            }
        }
        else if(opt==2)root[i]=root[x];
        else if(opt==3)
        {
            read(y);
            root[i]=root[i-1];
            static int posx,posy;
            posx=find(root[i],x);posy=find(root[i],y);
            if(fa[posx]==fa[posy])puts("1");
            else puts("0");
        }
    }
    return 0;
}
```

3.15 kd-tree

建树，每次选择方差最大的维度划分，复杂度 O(nlogn)
以最近点对查询为例
1. 最近点对即找到和当前点不同的其他点使其距离最短，复杂度 O(n)
2. 加点优化，记录子树表示的超长方体区域，若某一维的坐标距离大于当前答案则不遍历该子树
3. 启发式搜索，优先搜索坐标距离最大值最小的子树
复杂度最差还是 O(n)

3.15.1 方差建树

```
//n个点求距离小于dis的的点对个数。
#include <bits/stdc++.h>
using namespace std;
#define D 3
#define N 200010
const int inf=1000000001;
const long long Inf=1ll*inf*inf;
struct kdnode
{
    int x[D];
    int split;
    int l,r,p;
} kdtree[N],q[N];
bool operator==(const kdnode &a,const kdnode &b)
{
    for(int i=0; i<D; i++)
    {
        if(a.x[i]!=b.x[i])return false;
    }
    return true;
}
double avg[D],var[D];
```

```
int n, dis;
void calAvg(int l,int r)
{
   for(int i=0; i<D; i++)avg[i]=0;
   for(int i=l; i<=r; i++)
   for(int j=0; j<D; j++)
   avg[j]+=1.0*kdtree[i].x[j]/(r-l+1);
}
void calVar(int l,int r)
{
   for(int i=0; i<D; i++)var[i]=0;
   for(int i=l; i<=r; i++)
   for(int j=0; j<D; j++)
   var[j]+=1.0*(kdtree[i].x[j]-avg[j])/n*(kdtree[i].x[j]-avg[j]);
}
int splitD;
double maxVar;
bool cmp(kdnode a,kdnode b)
{
   return a.x[splitD]<b.x[splitD];
}
int construct(int p,int l,int r)//普通建树
{
   if(r<l)return -1;
   int root=(l+r)/2;
   calAvg(l,r);
   calVar(l,r);
   maxVar=-1.0;
   for(int i=0; i<D; i++)
   if(var[i]>maxVar)
   maxVar=var[i],splitD=i;
   sort(kdtree+l,kdtree+r+1,cmp);
   kdtree[root].split=splitD;
   kdtree[root].l=construct(root,l,root-1);
   kdtree[root].r=construct(root,root+1,r);
   kdtree[root].p=p;
   return root;
}

long long dist(kdnode a,kdnode b)
{
   long long ret=0;
   for(int i=0; i<D; i++)
   {
      ret+=1ll*(a.x[i]-b.x[i])*(a.x[i]-b.x[i]);
      if(ret > 1ll*dis*dis) return ret;
   }
   return ret;
}
int query(int root,kdnode x)
{
   if(root==-1)return 0;
   int d=kdtree[root].split;
   int ret = 0;
   if(x.x[d]<kdtree[root].x[d])
   {
      ret+=query(kdtree[root].l,x);
      if(x.x[d]+dis>kdtree[root].x[d])
      {
         ret+=query(kdtree[root].r,x);
      }
   }
   else if(x.x[d]>kdtree[root].x[d])
   {
      ret+=query(kdtree[root].r,x);
```

```
        if(x.x[d]-dis<kdtree[root].x[d])
        {
            ret+=query(kdtree[root].l,x);
        }
    }
    else
    {
        ret+=query(kdtree[root].l,x);
        ret+=query(kdtree[root].r,x);
    }
    if(dist(x, kdtree[root]) < 1ll*dis*dis) ret ++;
    return ret;
}
int main()
{
    //freopen("stars.in","r",stdin);
    while(scanf("%d%d",&n, &dis), n + dis)
    {
        for(int i=0; i<n; i++)
        {
            kdtree[i].split=0;
            kdtree[i].p=kdtree[i].l=kdtree[i].r=-1;
            for(int j=0; j<D; j++)
            {
                scanf("%d",&kdtree[i].x[j]);
                q[i]=kdtree[i];
            }
        }

        int root=construct(-1,0,n-1);
        int ans = 0;
        for(int i=0; i<n; i++)
        {
            ans += query(root,q[i]) - 1;
        }
        printf("%d\n", ans / 2);
    }
    return 0;
}
```

3.15.2 带修改

```
/*
hdu 2966 In case of failure 求离每个点最近点的距离。
方差建树。。
有插入删除操作（可以不用插入删除）
*/
#include <bits/stdc++.h>
using namespace std;
#define D 2
#define N 200010
const int inf=1000000001;
const long long Inf=1ll*inf*inf;
struct kdnode
{
    int x[D];
    int split;
    int l,r,p;
} kdtree[N],q[N];
bool operator==(const kdnode &a,const kdnode &b)
{
    for(int i=0; i<D; i++)
    {
        if(a.x[i]!=b.x[i])return false;
    }
    return true;
}
double avg[D],var[D];
int n;
void calAvg(int l,int r)
{
    for(int i=0; i<D; i++)avg[i]=0;
    for(int i=l; i<=r; i++)
    for(int j=0; j<D; j++)
    avg[j]+=1.0*kdtree[i].x[j]/(r-l+1);
}
void calVar(int l,int r)
{
    for(int i=0; i<D; i++)var[i]=0;
    for(int i=l; i<=r; i++)
    for(int j=0; j<D; j++)
    var[j]+=1.0*(kdtree[i].x[j]-avg[j])/n*(kdtree[i].x[j]-avg[j]);
}
int splitD;
double maxVar;
bool cmp(kdnode a,kdnode b)
{
    return a.x[splitD]<b.x[splitD];
}
int construct(int p,int l,int r)
{
    if(r<l)return -1;
    int root=(l+r)/2;
    calAvg(l,r);
    calVar(l,r);
    maxVar=-1.0;
    for(int i=0; i<D; i++)
    if(var[i]>maxVar)
    maxVar=var[i],splitD=i;
    sort(kdtree+l,kdtree+r+1,cmp);
    kdtree[root].split=splitD;
    kdtree[root].l=construct(root,l,root-1);
    kdtree[root].r=construct(root,root+1,r);
    kdtree[root].p=p;
    return root;
}
int Find(int root,kdnode x)
{
    if(root==-1)return -1;
    if(x==kdtree[root])
    {
        return root;
    }
    int d=kdtree[root].split;
    if(x.x[d]>kdtree[root].x[d])
    {
        return Find(kdtree[root].r,x);
    }
    else if(x.x[d]<kdtree[root].x[d])
    {
        return Find(kdtree[root].l,x);
    }
    else
    {
        int l=Find(kdtree[root].l,x);
        int r=Find(kdtree[root].r,x);
        return (l==-1?r:l);
    }
}
int FindMin(int root,int d)
```

```
{
   int ret=root;
   if(kdtree[root].l!=-1)
   {
      int v=FindMin(kdtree[root].l,d);
      if(kdtree[ret].x[d]>kdtree[v].x[d])
      ret=v;
   }
   if(kdtree[root].r!=-1)
   {
      int v=FindMin(kdtree[root].r,d);
      if(kdtree[ret].x[d]>kdtree[v].x[d])
      ret=v;
   }
   return ret;
}
int FindMax(int root,int d)
{
   int ret=root;
   if(kdtree[root].l!=-1)
   {
      int v=FindMax(kdtree[root].l,d);
      if(kdtree[ret].x[d]<kdtree[v].x[d])
      ret=v;
   }
   if(kdtree[root].r!=-1)
   {
      int v=FindMax(kdtree[root].r,d);
      if(kdtree[ret].x[d]<kdtree[v].x[d])
      ret=v;
   }
   return ret;
}
void DeleteNode(int v)
{
   int p=kdtree[v].p;
   kdtree[v].p=-1;
   if(kdtree[p].l==v)
   kdtree[p].l=-1;
   else
   kdtree[p].r=-1;
}
void Remove(int root,kdnode x)
{
   int pos=Find(root,x);
   if(kdtree[pos].l==-1&&kdtree[pos].r==-1)
   {
      DeleteNode(pos);
   }
   else if(kdtree[pos].l==-1)
   {
      int alt=FindMin(kdtree[pos].r,kdtree[pos].split);
      for(int i=0; i<D; i++)kdtree[pos].x[i]=kdtree[alt].x[i];
      Remove(alt,kdtree[alt]);
   }
   else
   {
      int alt=FindMax(kdtree[pos].l,kdtree[pos].split);
      for(int i=0; i<D; i++)kdtree[pos].x[i]=kdtree[alt].x[i];
      Remove(alt,kdtree[alt]);
   }
}
void Insert(int root,int x)
{
   int d=kdtree[root].split;
```

```
    if(kdtree[root].x[d]<kdtree[x].x[d])
    {
        if(kdtree[root].r==-1)
        {
            kdtree[root].r=x;
            kdtree[x].p=root;
        }
        else Insert(kdtree[root].r,x);
    }
    else
    {
        if(kdtree[root].l==-1)
        {
            kdtree[root].l=x;
            kdtree[x].p=root;
        }
        else Insert(kdtree[root].l,x);
    }
}
void Add(int root,kdnode x)
{
    int pos=n;
    kdtree[n++]=x;
    Insert(root,pos);
}

long long dist(kdnode a,kdnode b)
{
    long long ret=0;
    for(int i=0; i<D; i++)
    {
        ret+=1ll*(a.x[i]-b.x[i])*(a.x[i]-b.x[i]);
    }
    return ret;
}
long long query(int root,kdnode x)
{
    if(root==-1)return Inf;
    int d=kdtree[root].split;
    long long ret;
    if(x.x[d]<kdtree[root].x[d])
    {
        ret=query(kdtree[root].l,x);
        double dd=1.0*x.x[d]+sqrt(1.0*ret);
        if(dd>=1.0*kdtree[root].x[d])
        {
            ret=min(ret,query(kdtree[root].r,x));
        }
    }
    else if(x.x[d]>kdtree[root].x[d])
    {
        ret=query(kdtree[root].r,x);
        double dd=1.0*x.x[d]-sqrt(1.0*ret);
        if(dd<=1.0*kdtree[root].x[d])
        {
            ret=min(ret,query(kdtree[root].l,x));
        }
    }
    else
    {
        ret=query(kdtree[root].l,x);
        ret=min(ret,query(kdtree[root].r,x));
    }
    ret=min(ret,dist(kdtree[root],x));
    return ret;
```

```
}
int main()
{
    // freopen("in","r",stdin);
    int t;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d",&n);
        for(int i=0; i<n; i++)
        {
            kdtree[i].split=0;
            kdtree[i].p=kdtree[i].l=kdtree[i].r=-1;
            for(int j=0; j<D; j++)
            {
                scanf("%d",&kdtree[i].x[j]);
                q[i]=kdtree[i];
            }
        }

        int root=construct(-1,0,n-1);

        int m=n;
        for(int i=0; i<m; i++)
        {
            Remove(root,q[i]);
            cout<<query(root,q[i])<<endl;
            Add(root,q[i]);
        }
    }
    return 0;
}
```

3.15.3 矩形区域和

```
//强制在线+卡空间
//维度k，查询复杂度最优为logn，最差为n^{1-1/k}
//同样是记录子树的超长方体区域，不相交则不遍历，完全包含则直接加上子树和，否则继续遍历
#include <bits/stdc++.h>
using namespace std;
const int maxn = 200010;
int n, op, xl, xr, yl, yr, lstans;
struct node {
    int x, y, v;
} s[maxn];
bool cmp1(int a, int b) { return s[a].x < s[b].x; }
bool cmp2(int a, int b) { return s[a].y < s[b].y; }
double a = 0.725;
int rt, cur, d[maxn], lc[maxn], rc[maxn], L[maxn], R[maxn], D[maxn], U[maxn],
siz[maxn], sum[maxn];
int g[maxn], t;
void print(int x) {
    if (!x) return;
    print(lc[x]);
    g[++t] = x;
    print(rc[x]);
}
void maintain(int x) {
    siz[x] = siz[lc[x]] + siz[rc[x]] + 1;
    sum[x] = sum[lc[x]] + sum[rc[x]] + s[x].v;
    L[x] = R[x] = s[x].x;
    D[x] = U[x] = s[x].y;
    if (lc[x])
    L[x] = min(L[x], L[lc[x]]), R[x] = max(R[x], R[lc[x]]),
    D[x] = min(D[x], D[lc[x]]), U[x] = max(U[x], U[lc[x]]);
    if (rc[x])
    L[x] = min(L[x], L[rc[x]]), R[x] = max(R[x], R[rc[x]]),
    D[x] = min(D[x], D[rc[x]]), U[x] = max(U[x], U[rc[x]]);
}
int build(int l, int r) {
    if (l > r) return 0;
    int mid = (l + r) >> 1;
    double av1 = 0, av2 = 0, va1 = 0, va2 = 0;
    for (int i = l; i <= r; i++) av1 += s[g[i]].x, av2 += s[g[i]].y;
    av1 /= (r - l + 1);
    av2 /= (r - l + 1);
    for (int i = l; i <= r; i++)
    va1 += (av1 - s[g[i]].x) * (av1 - s[g[i]].x),
    va2 += (av2 - s[g[i]].y) * (av2 - s[g[i]].y);
    if (va1 > va2)
    nth_element(g + l, g + mid, g + r + 1, cmp1), d[g[mid]] = 1;
    else
    nth_element(g + l, g + mid, g + r + 1, cmp2), d[g[mid]] = 2;
    lc[g[mid]] = build(l, mid - 1);
    rc[g[mid]] = build(mid + 1, r);
    maintain(g[mid]);
    return g[mid];
}
void rebuild(int& x) {
    t = 0;
    print(x);
    x = build(1, t);
}
bool bad(int x) { return a * siz[x] <= (double)max(siz[lc[x]], siz[rc[x]]); }
void insert(int& x, int v) {
    if (!x) {
        x = v;
        maintain(x);
        return;
    }
    if (d[x] == 1) {
        if (s[v].x <= s[x].x)
        insert(lc[x], v);
        else
        insert(rc[x], v);
    } else {
        if (s[v].y <= s[x].y)
        insert(lc[x], v);
        else
        insert(rc[x], v);
    }
    maintain(x);
    if (bad(x)) rebuild(x);
}
int query(int x) {
    if (!x || xr < L[x] || xl > R[x] || yr < D[x] || yl > U[x]) return 0;
    if (xl <= L[x] && R[x] <= xr && yl <= D[x] && U[x] <= yr) return sum[x];
    int ret = 0;
    if (xl <= s[x].x && s[x].x <= xr && yl <= s[x].y && s[x].y <= yr)
    ret += s[x].v;
    return query(lc[x]) + query(rc[x]) + ret;
}
int main() {
    scanf("%d", &n);
    while (~scanf("%d", &op)) {
        if (op == 1) {
            cur++, scanf("%d%d%d", &s[cur].x, &s[cur].y, &s[cur].v);
            s[cur].x ^= lstans;
            s[cur].y ^= lstans;
            s[cur].v ^= lstans;
            insert(rt, cur);
        }
        if (op == 2) {
            scanf("%d%d%d%d", &xl, &yl, &xr, &yr);
            xl ^= lstans;
            yl ^= lstans;
            xr ^= lstans;
            yr ^= lstans;
            printf("%d\n", lstans = query(rt));
        }
        if (op == 3) return 0;
    }
}
```

3.16 支配树

```
/*
建树O(cn)
它是一棵树，根节点是我们选定的起点s。
对于每个点i，它到根的链上的点集就是对于它的必经点集{xi}。
对于每个点i，它是它的支配树上的子树内的点的必经点。
*/
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int N = 100010;
const int M = 500010;
int n,m,fa[N],dfn[N],rev[N],clo,semi[N],idom[N],size[N];

inline int gi(){
    int x=0,res=1;char ch=getchar();
    while(ch>'9' || ch<'0')res^=ch=='-',ch=getchar();
    while(ch>='0'&&ch<='9')x=x*10+ch-48,ch=getchar();
    return res?x:-x;
}

struct Node{int to,next;};
struct Graph{
    Node E[M];int head[N],tot;
    inline void clr(){
        for(int i=tot=0;i<=n;++i)head[i]=0;
    }
    inline void link(int u,int v){
        E[++tot]=(Node){v,head[u]};
        head[u]=tot;
    }
}pre,nxt,dom;

struct Union_Merge_Set{
    int fa[N],Mi[N];
    inline void init(){
        for(int i=1;i<=n;++i)
        fa[i]=Mi[i]=semi[i]=i;
    }
    inline int find(int x){
        if(fa[x]==x)return x;
        int fx=fa[x],y=find(fa[x]);
        if(dfn[semi[Mi[fx]]]<dfn[semi[Mi[x]]])Mi[x]=Mi[fx];
        return fa[x]=y;
    }
}uset;

inline void tarjan(int x){
    dfn[x]=++clo;rev[clo]=x;
    for(int e=nxt.head[x];e;e=nxt.E[e].next)
    if(!dfn[nxt.E[e].to])
    fa[nxt.E[e].to]=x,tarjan(nxt.E[e].to);
}

inline void build(){
    for(int i=n;i>=2;--i){
        int y=rev[i],tmp=n;if(!y)continue;
        for(int e=pre.head[y];e;e=pre.E[e].next){
            int x=pre.E[e].to;if(!dfn[x])continue;
            if(dfn[x]<dfn[y])tmp=min(tmp,dfn[x]);
            else uset.find(x),tmp=min(tmp,dfn[semi[uset.Mi[x]]]);
        }
        semi[y]=rev[tmp];uset.fa[y]=fa[y];
        dom.link(semi[y],y);

        y=rev[i-1];if(!y)continue;
        for(int e=dom.head[y];e;e=dom.E[e].next){
            int x=dom.E[e].to;uset.find(x);
            if(semi[uset.Mi[x]]==y)idom[x]=y;
            else idom[x]=uset.Mi[x];
        }
    }
    for(int i=2;i<=n;++i){
        int x=rev[i];
        if(idom[x]!=semi[x])
        idom[x]=idom[idom[x]];
    }
    dom.clr();

    for(int i=2;i<=n;++i)
    dom.link(idom[rev[i]],rev[i]);
}

inline void dfs(int x){
    size[x]=1;
    for(int e=dom.head[x];e;e=dom.E[e].next){
        int y=dom.E[e].to;if(size[y])continue;
        dfs(y);size[x]+=size[y];
    }
}

inline LL calc(LL Ans=0,LL sum=0){
    for(int e=dom.head[1];e;e=dom.E[e].next){
        int y=dom.E[e].to;
        Ans+=sum*size[y];
        sum+=size[y];
    }
    return Ans+size[1]-1;
}

int main(){
    n=gi();m=gi();
    for(int i=1;i<=m;++i){
        int u=gi(),v=gi();
        nxt.link(u,v);
        pre.link(v,u);
    }
    tarjan(1);
    uset.init();
    build();
    dfs(1);
    printf("%lld",calc());
    return 0;
}
```

3.17 笛卡尔树

```
#include <iostream>
using namespace std;
typedef long long ll;
const int N = 100000 + 10, INF = 0x3f3f3f3f;

struct node {
   int idx, val, par, ch[2];
   friend bool operator<(node a, node b) { return a.idx < b.idx; }
   void init(int _idx, int _val, int _par) {
      idx = _idx, val = _val, par = _par, ch[0] = ch[1] = 0;
   }
} tree[N];

int root, top, stk[N];
ll ans;
int cartesian_build(int n) {
   for (int i = 1; i <= n; i++) {
      int k = i - 1;
      while (tree[k].val > tree[i].val) k = tree[k].par;
      tree[i].ch[0] = tree[k].ch[1];
      tree[k].ch[1] = i;
      tree[i].par = k;
      tree[tree[i].ch[0]].par = i;
   }
   return tree[0].ch[1];
}
int dfs(int x) {
   if (!x) return 0;
   int sz = dfs(tree[x].ch[0]);
   sz += dfs(tree[x].ch[1]);
   ans = max(ans, (ll)(sz + 1) * tree[x].val);
   return sz + 1;
}
int main() {
   int n, hi;
   while (scanf("%d", &n), n) {
   tree[0].init(0, 0, 0);
   for (int i = 1; i <= n; i++) {
      scanf("%d", &hi);
      tree[i].init(i, hi, 0);
   }
   root = cartesian_build(n);
   ans = 0;
   dfs(root);
   printf("%lld\n", ans);
}
return 0;
}
```

3.18 虚树

```
#include<bits/stdc++.h>
#define MAXV 100005
#define INF 1000000000
#define MAXLOGV 20
using namespace std;
struct edge
{
   int to,cost;
};
vector<edge> G[MAXV];
vector<int> vt[MAXV];
```

```
int parent[MAXLOGV][MAXV];
int depth[MAXV],dfn[MAXV],dis[MAXV],st[MAXV];
int n,q,tot;
void add_edge(int from,int to)
{
   vt[from].push_back(to);
}
bool cmp(int x,int y)
{
   return dfn[x]<dfn[y];
}
void dfs(int v,int p,int d,int minx)
{
   dfn[v]=++tot;
   dis[v]=minx;
   parent[0][v]=p;
   depth[v]=d;
   for(int i=0;i<(int)G[v].size();i++)
   if(G[v][i].to!=p) dfs(G[v][i].to,v,d+1,min(minx,G[v][i].cost));
}
void init(int V)
{
   dfs(1,-1,0,INF);
   for(int k=0;k+1<MAXLOGV;k++)
   {
      for(int v=1;v<=V;v++)
      {
         if(parent[k][v]<0) parent[k+1][v]=-1;
         else parent[k+1][v]=parent[k][parent[k][v]];
      }
   }
}
int lca(int u,int v)
{
   if(depth[u]>depth[v]) swap(u,v);
   for(int k=0;k<MAXLOGV;k++)
   {
      if((depth[v]-depth[u])>>k&1)
      v=parent[k][v];
   }
   if(u==v) return u;
   for(int k=MAXLOGV-1;k>=0;k--)
   {
      if(parent[k][u]!=parent[k][v])
      {
         u=parent[k][u];
         v=parent[k][v];
      }
   }
   return parent[0][u];
}
int build_vtree(vector<int> &a)
{
   sort(a.begin(),a.end(),cmp);
   a.erase(unique(a.begin(),a.end()),a.end());
   assert(a.size()>0);
   int t=0;
   st[t++]=a[0];
   vector<int> newly;newly.clear();
   for(int i=1;i<(int)a.size();i++)
   {
      if(t==0) {st[t++]=a[i]; continue;}
      int l=lca(a[i],st[t-1]);
      while(t>1&&dfn[st[t-2]]>=dfn[l]) add_edge(st[t-2],st[t-1]),t--;
      if(l!=st[t-1]) {add_edge(l,st[t-1]),st[t-1]=l; newly.push_back(l);}
```

```
        st[t++]=a[i];
    }
    while(t>1) add_edge(st[t-2],st[t-1]),t--;
    for(auto it:newly) a.push_back(it);
    return st[0];
}
int main()
{
    return 0;
}
```

3.19 树套树

3.19.1 树状数组套主席树

```
/*
树上带修改第k大
结点维护从自身到根上的信息，查询时u+v-lca(u,v)-fa[lca(u,v)]
*/
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int N=16e4+5;
const int maxn=N;
int n,m;
int w[maxn],t[maxn],vnum;

struct Query{
    int k,a,b;
}q[maxn];


int addT[maxn],anum,subT[maxn],snum;
int T[maxn],L[maxn<<7],R[maxn<<7],sum[maxn<<7],tnum;
void update(int &rt,int l,int r,int x,int d){
    if(!rt)rt=++tnum;
    sum[rt]+=d;
    if(l<r){
        int mid=(l+r)/2;
        if(x<=mid)update(L[rt],l,mid,x,d);
        else update(R[rt],mid+1,r,x,d);
    }
}
int query(int l,int r,int k){
    if(l==r)return l;
    int tmp=0;
    for(int i=1;i<=anum;i++)tmp+=sum[R[addT[i]]];
    for(int i=1;i<=snum;i++)tmp-=sum[R[subT[i]]];
    int mid=(l+r)/2;
    if(k<=tmp){
        for(int i=1;i<=anum;i++)addT[i]=R[addT[i]];
        for(int i=1;i<=snum;i++)subT[i]=R[subT[i]];
        return query(mid+1,r,k);
    }
    else{
        for(int i=1;i<=anum;i++)addT[i]=L[addT[i]];
        for(int i=1;i<=snum;i++)subT[i]=L[subT[i]];
        return query(l,mid,k-tmp);
    }
}
```

```
int lowbit(int x){return x&(-x);}
void add(int x,int val,int d){
    for(;x<=n;x+=lowbit(x))
    update(T[x],1,vnum,val,d);
}



int head[maxn],tot=1;
struct Edge{
    int v,nxt;
}e[maxn<<1];
void addedge(int u,int v){
    e[tot].v=v;e[tot].nxt=head[u];head[u]=tot++;
    e[tot].v=u;e[tot].nxt=head[v];head[v]=tot++;
}
int sz[maxn],son[maxn],fa[maxn],h[maxn],pos[maxn],top[maxn],low[maxn],cnt;
void dfs1(int u,int f){
    sz[u]=1;son[u]=0;fa[u]=f;h[u]=h[f]+1;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(v==f)continue;
        dfs1(v,u);
        sz[u]+=sz[v];
        if(sz[son[u]]<sz[v])son[u]=v;
    }
}
void dfs2(int u,int f,int k){
    top[u]=k;pos[u]=++cnt;
    if(son[u])dfs2(son[u],u,k);
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(v==f || v==son[u])continue;
        dfs2(v,u,v);
    }
    low[u]=cnt;
}
int LCA(int u,int v){
    while(top[u]!=top[v]){
        if(h[top[u]]<h[top[v]])swap(u,v);
        u=fa[top[u]];
    }
    if(h[u]>h[v])swap(u,v);
    return u;
}



int main()
{
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++)scanf("%d",&w[i]),t[++vnum]=w[i];
    int u,v,k;
    for(int i=1;i<=n-1;i++){
        scanf("%d %d",&u,&v);
        addedge(u,v);
    }
    for(int i=1;i<=m;i++){
        scanf("%d %d %d",&q[i].k,&q[i].a,&q[i].b);
        if(!q[i].k)t[++vnum]=q[i].b;
    }
    sort(t+1,t+1+vnum);
    vnum=unique(t+1,t+1+vnum)-(t+1);
    for(int i=1;i<=n;i++)w[i]=lower_bound(t+1,t+1+vnum,w[i])-t;
    for(int i=1;i<=m;i++)if(!q[i].k)q[i].b=lower_bound(t+1,t+1+vnum,q[i].b)-t;
    dfs1(1,0);dfs2(1,0,1);


    for(int i=1;i<=n;i++)add(pos[i],w[i],1),add(low[i]+1,w[i],-1);


    for(int i=1;i<=m;i++){
        u=q[i].a;v=q[i].b;k=q[i].k;
        if(k){
            int lca=LCA(u,v);
            if(h[u]+h[v]-h[lca]+-h[fa[lca]]<k){
                printf("invalid request!\n");
                continue;
            }
            anum=snum=0;
            for(int i=pos[u];i;i-=lowbit(i))addT[++anum]=T[i];
            for(int i=pos[v];i;i-=lowbit(i))addT[++anum]=T[i];
            for(int i=pos[lca];i;i-=lowbit(i))subT[++snum]=T[i];
            for(int i=pos[fa[lca]];i;i-=lowbit(i))subT[++snum]=T[i];
            printf("%d\n",t[query(1,vnum,k)]);
        }
        else{
            add(pos[u],w[u],-1);add(low[u]+1,w[u],1);
            w[u]=v;
            add(pos[u],w[u],1);add(low[u]+1,w[u],-1);
        }
    }
    return 0;
}
```

3.19.2 线段树套平衡树

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int N=5e4+5;
const int maxn=N<<6;
const int maxv=1e8+5;
const int INF=1e9+9;

inline int read(){
    char ch=getchar();
    int res=0,f=1;
    while(!isdigit(ch)){if(ch=='-')f=-f;ch=getchar();}
    while(isdigit(ch))res=(res<<3)+(res<<1)+(ch^48),ch=getchar();
    return res*f;
}


int n,m;
int c[maxn],ma;

namespace treap
{
    struct node
    {
        int l,r,v,rnd,sz;
    } tr[maxn];
    int tot,root;
```

```
inline void init()
{
   tot=0;
   root=0;
}

inline void update(int x)
{
   tr[x].sz=tr[tr[x].l].sz+tr[tr[x].r].sz+1;
}

inline int newnode(int v)
{
   ++tot;
   tr[tot].sz=1;tr[tot].v=v;tr[tot].rnd=rand();
   tr[tot].l=tr[tot].r=0;
   return tot;
}

int merge(int x,int y)//x<y
{
   if(!x || !y)return x+y;
   if(tr[x].rnd<tr[y].rnd)
   {
      tr[x].r=merge(tr[x].r,y);
      update(x);
      return x;
   }
   else
   {
      tr[y].l=merge(x,tr[y].l);
      update(y);
      return y;
   }
}

void split(int now,int k,int &x,int &y){
   if(!now)x=y=0;
   else
   {
      if(tr[now].v<=k)
      {
         x=now;
         split(tr[now].r,k,tr[now].r,y);
      }
      else
      {
         y=now;
         split(tr[now].l,k,x,tr[now].l);
      }
      update(now);
   }
}

inline void Insert(int &rt,int v)
{
   int x,y;
   split(rt,v,x,y);
   rt=merge(merge(x,newnode(v)),y);
}

inline void Delete(int &rt,int v)
{
   int x1,y1,x2,y2;
   split(rt,v,x1,y1);
```

```
        split(x1,v-1,x2,y2);
        y2=merge(tr[y2].l,tr[y2].r);
        rt=merge(merge(x2,y2),y1);
    }

    inline int getrank(int &rt,int v)
    {
        int x,y;
        split(rt,v-1,x,y);
        int res=tr[x].sz;
        rt=merge(x,y);
        return res;
    }

    inline int getpre(int &rt,int v)
    {
        int x,y;
        split(rt,v-1,x,y);
        if(x==0)return -INF;
        int res=x;
        while(tr[res].r)res=tr[res].r;
        rt=merge(x,y);
        return tr[res].v;
    }

    inline int getsuc(int &rt,int v)
    {
        int x,y;
        split(rt,v,x,y);
        if(y==0)return INF;
        int res=y;
        while(tr[res].l)res=tr[res].l;
        rt=merge(x,y);
        return tr[res].v;
    }
}



namespace segtree{
    int rt[maxn];

    void build(int x,int l,int r){
        for(int i=l;i<=r;++i)treap::Insert(rt[x],c[i]);
        if(l==r)return;
        int mid=(l+r)/2;
        build(x<<1,l,mid);build(x<<1|1,mid+1,r);
    }

    void update(int x,int l,int r,int pos,int PV,int V){
        treap::Delete(rt[x],PV);
        treap::Insert(rt[x],V);
        if(l==r)return;
        int mid=(l+r)/2;
        if(pos<=mid)update(x<<1,l,mid,pos,PV,V);
        else if(pos>mid)update(x<<1|1,mid+1,r,pos,PV,V);
    }

    int getrank(int x,int l,int r,int L,int R,int V){
        if(l==L && r==R)return treap::getrank(rt[x],V);
        int mid=(l+r)/2;
        if(R<=mid)return getrank(x<<1,l,mid,L,R,V);
        else if(L>mid)return getrank(x<<1|1,mid+1,r,L,R,V);
        else return getrank(x<<1,l,mid,L,mid,V)+getrank(x<<1|1,mid+1,r,mid+1,R,V);
    }

    int getpre(int x,int l,int r,int L,int R,int V){
        if(l==L && r==R)return treap::getpre(rt[x],V);
        int mid=(l+r)/2;
        if(R<=mid)return getpre(x<<1,l,mid,L,R,V);
        else if(L>mid)return getpre(x<<1|1,mid+1,r,L,R,V);
        else return max(getpre(x<<1,l,mid,L,mid,V),getpre(x<<1|1,mid+1,r,mid+1,R,V));
    }

    int getsuc(int x,int l,int r,int L,int R,int V){
        if(l==L && r==R)return treap::getsuc(rt[x],V);
        int mid=(l+r)/2;
        if(R<=mid)return getsuc(x<<1,l,mid,L,R,V);
        else if(L>mid)return getsuc(x<<1|1,mid+1,r,L,R,V);
        else return min(getsuc(x<<1,l,mid,L,mid,V),getsuc(x<<1|1,mid+1,r,mid+1,R,V));
    }
}




inline void qrank(){
    int ans=0,l,r,k;
    l=read();r=read();k=read();
    ans=segtree::getrank(1,1,n,l,r,k)+1;
    printf("%d\n",ans);
}

inline void qkth(){
    int l,r,k;
    l=read();r=read();k=read();
    int L=0,R=ma+1;
    while(L<R){
        int mid=(L+R)/2;
        int tmp=segtree::getrank(1,1,n,l,r,mid);
        if(tmp>=k)R=mid;
        else L=mid+1;
    }
    printf("%d\n",L-1);
}

inline void modify(){
    int pos,k;
    pos=read();k=read();
    segtree::update(1,1,n,pos,c[pos],k);
    c[pos]=k;
}

inline void qpre(){
    int ans=0,l,r,k;
    l=read();r=read();k=read();
    ans=segtree::getpre(1,1,n,l,r,k);
    printf("%d\n",ans);
}

inline void qsuc(){
    int ans=0,l,r,k;
    l=read();r=read();k=read();
    ans=segtree::getsuc(1,1,n,l,r,k);
    printf("%d\n",ans);
}


int main()
```

```
{
    n=read();m=read();
    ma=-1;
    treap::init();
    for(int i=1;i<=n;++i)c[i]=read(),ma=max(ma,c[i]);
    segtree::build(1,1,n);

    int op,x,y;
    for(int i=1;i<=m;++i){
        op=read();
        switch(op)
        {
            case 1:qrank();break;
            case 2:qkth();break;
            case 3:modify();break;
            case 4:qpre();break;
            case 5:qsuc();break;
        }
    }
    return 0;
}
```

3.19.3 平衡树套主席树

```
//带单点添加的第k大
#include <bits/stdc++.h>
#define debug(...) fprintf(stderr, __VA_ARGS__)
using namespace std;

template<class T>
inline void read(T &x)
{
    char c;int f = 1;x = 0;
    while(((c=getchar()) < '0' || c > '9') && c != '-');
    if(c == '-') f = -1;else x = c-'0';
    while((c=getchar()) >= '0' && c <= '9') x = x*10+c-'0';
    x *= f;
}

int OutN;
char Out[20];

template<class T>
inline void write(T x)
{
    if(x < 0) putchar('-'), x = -x;
    if(x)
    {
        OutN = 0;
        while(x)
        {
            Out[OutN++] = x%10+'0';
            x /= 10;
        }
        while(OutN--)
        putchar(Out[OutN]);
    }
    else putchar('0');
}

const int N = 70009;
const int V = 70000;
const int LOGN = 20;
const int SZ = 20000000;
const double ALPHA = 0.75;
const double LOGALPHA = log(4.0)-log(3.0);

int n, v[N];

namespace SegmentTree
{
   int lc[SZ], rc[SZ], sum[SZ];
   int Stack[SZ], Stop;

   void init(int x)
   {
      lc[x] = rc[x] = sum[x] = 0;
   }

   int newNode()
   {
      int ret = Stack[Stop--];
      init(ret);return ret;
   }

   void insert(int &x, int l, int r, int pos, int v)
   {
      if(x == 0)
      x = newNode();
      sum[x] += v;
      if(l == r)
      return ;
      int mid = (l+r)>>1;
      if(pos <= mid) insert(lc[x], l, mid, pos, v);
      else insert(rc[x], mid+1, r, pos, v);
   }

   void newTree(int &x, int pos)
   {
      x = newNode(), sum[x]++;
      int l = 0, r = V, cur = x;
      while(l < r)
      {
         int mid = (l+r)>>1;
         if(pos <= mid) cur = lc[cur] = newNode(), r = mid;
         else cur = rc[cur] = newNode(), l = mid+1;
         sum[cur]++;
      }
   }

   void merge(int &x, int y)
   {
      if(y == 0)
      return ;
      if(x == 0)
      x = newNode();
      sum[x] += sum[y];
      merge(lc[x], lc[y]);
      merge(rc[x], rc[y]);
   }

   void clear(int &x)
   {
      Stack[++Stop] = x;
      if(lc[x]) clear(lc[x]);
      if(rc[x]) clear(rc[x]);
      init(x), x = 0;
   }
   /*
   void Debug(int x, int l, int r)
```

```
    {
        if(l == r)
        {
            debug("%d : %d\n", l, sum[x]);
            return ;
        }
        int mid = (l+r)>>1;
        if(lc[x]) Debug(lc[x], l, mid);
        if(rc[x]) Debug(rc[x], mid+1, r);
    }*/
}

namespace ScapgoatTree
{
    int root;
    int lc[N], rc[N], sz[N], rt[N];
    int dfn[N], dfnN, val[N], valN;
    int maxDepth;

    int build(int l, int r)
    {
        int mid = (l+r)>>1, x = dfn[mid];
        SegmentTree::newTree(rt[x], v[x]);
        if(l < mid) lc[x] = build(l, mid-1);
        if(mid < r) rc[x] = build(mid+1, r);
        sz[x] = sz[lc[x]]+sz[rc[x]]+1;
        SegmentTree::merge(rt[x], rt[lc[x]]);
        SegmentTree::merge(rt[x], rt[rc[x]]);
        return x;
    }

    void get(int x)
    {
        if(lc[x]) get(lc[x]);
        dfn[++dfnN] = x;
        if(rc[x]) get(rc[x]);
    }

    int rebuild(int x)
    {
        dfnN = 0, get(x);
        for(int i = 1; i <= dfnN; ++i)
        {
            SegmentTree::clear(rt[dfn[i]]);
            lc[dfn[i]] = rc[dfn[i]] = sz[dfn[i]] = 0;
        }
        return build(1, dfnN);
    }

    void get(int x, int l, int r, int ql, int qr)
    {
        if(ql <= l && r <= qr)
        {
            dfn[++dfnN] = rt[x];
            return ;
        }
        int mid = l+sz[lc[x]];
        if(ql < mid && lc[x]) get(lc[x], l, mid-1, ql, qr);
        if(ql <= mid && mid <= qr) val[++valN] = v[x];
        if(qr > mid && rc[x]) get(rc[x], mid+1, r, ql, qr);
    }

    int query(int l, int r, int k)
    {
        valN = dfnN = 0, get(root, 1, n, l, r);
        l = 0, r = V;
        while(l < r)
        {
            int ls = 0, mid = (l+r)>>1;
            for(int i = 1; i <= dfnN; ++i)
            ls += SegmentTree::sum[SegmentTree::lc[dfn[i]]];
            for(int i = 1; i <= valN; ++i)
            if(l <= val[i] && val[i] <= mid) ls++;
            if(ls >= k)
            {
                r = mid;
                for(int i = 1; i <= dfnN; ++i)
                dfn[i] = SegmentTree::lc[dfn[i]];
            }
            else
            {
                k -= ls, l = mid+1;
                for(int i = 1; i <= dfnN; ++i)
                dfn[i] = SegmentTree::rc[dfn[i]];
            }
        }
        return l;
    }

    void get(int x, int pos)
    {
        dfn[++dfnN] = x;
        if(sz[lc[x]] >= pos) get(lc[x], pos);
        else if(sz[lc[x]]+1 == pos) return ;
        else get(rc[x], pos-sz[lc[x]]-1);
    }

    void modify(int x, int value)
    {
        dfnN = 0, get(root, x);
        for(int i = 1; i <= dfnN; ++i)
        {
            SegmentTree::insert(rt[dfn[i]], 0, V, v[dfn[dfnN]], -1);
            SegmentTree::insert(rt[dfn[i]], 0, V, value, 1);
        }
        v[dfn[dfnN]] = value;
    }

    bool insert(int &x, int pos, int p, int d)
    {
        if(x == 0)
        {
            sz[x = p]++;
            SegmentTree::newTree(rt[x], v[x]);
            return d <= maxDepth;
        }
        sz[x]++;
        SegmentTree::insert(rt[x], 0, V, v[p], 1);
        bool ret;
        if(pos <= sz[lc[x]]+1) ret = insert(lc[x], pos, p, d+1);
        else ret = insert(rc[x], pos-sz[lc[x]]-1, p, d+1);
        int lim = (int)(sz[x]*ALPHA);
        if(ret && (sz[lc[x]] > lim || sz[rc[x]] > lim))
        {
            x = rebuild(x);
            return false;
        }
        else return ret;
    }
    void insert(int pos, int p)
    {
        maxDepth = log(1.0*n)/LOGALPHA;
        insert(root, pos, p, 0);
    }
    /*
    void Debug(int x)
    {
        if(lc[x]) Debug(lc[x]);
        debug("%d ", v[x]);
        if(rc[x]) Debug(rc[x]);
    }*/
}

void init()
{
    read(n);
    for(int i = 1; i <= n; ++i)
    {
        read(v[i]);
        ScapgoatTree::dfn[i] = i;
    }
    for(int i = 1; i < SZ; ++i)
    SegmentTree::Stack[++SegmentTree::Stop] = SZ-i;
    ScapgoatTree::root = ScapgoatTree::build(1, n);
}

void solve()
{
    int qN;
    read(qN);
    int lastAns = 0;
    while(qN--)
    {
        int x, y, z;
        char str[4] = "\0";
        scanf("%s", str);
        switch(str[0])
        {
            case 'Q':
            read(x), read(y), read(z);
            x ^= lastAns, y ^= lastAns, z ^= lastAns;
            write(lastAns = ScapgoatTree::query(x, y, z)), putchar('\n');
            break;
            case 'M':
            read(x), read(y);
            x ^= lastAns, y ^= lastAns;
            ScapgoatTree::modify(x, y);
            break;
            default:
            read(x), read(y);
            x ^= lastAns, y ^= lastAns, v[++n] = y;
            ScapgoatTree::insert(x, n);
        }
    }
}

int main()
{
    freopen("bzoj3065.in", "r", stdin);
    freopen("bzoj3065.out", "w", stdout);

    init();
    solve();
    fclose(stdin);fclose(stdout);
    return 0;
}
```

4 图

4.1 SPFA

```
#include<bits/stdc++.h>
using namespace std;
const int INF=0x3f3f3f3f;
const int maxn=1e5+10;
int n,tot,S,T,u,v,w;
int dis[maxn],first[maxn],vis[maxn];
struct Node{
    int v,w,net;
} edge[maxn*10];

void addedge(int u,int v,int w)
{
    edge[tot].v=v;
    edge[tot].w=w;
    edge[tot].net=first[u];
    first[u]=tot++;
}

void SPFA(int s)
{
    queue<int> q;
    memset(dis,-INF,sizeof dis);
    memset(vis,0,sizeof vis);
    q.push(s); dis[s]=0;
    while(!q.empty())
    {
        int u=q.front(); q.pop();
        vis[u]=0;
        for(int i=first[u];~i;i=edge[i].net)
        {
            if(dis[edge[i].v]<dis[u]+edge[i].w)
            {
                dis[edge[i].v]=dis[u]+edge[i].w;
                if(!vis[edge[i].v])
                {
                    vis[edge[i].v]=1;
                    q.push(edge[i].v);
                }
            }
        }
    }
}
int main()
{
    while(~scanf("%d",&n))
    {
        tot=1, S=INF,T=-INF;
        memset(first,-1,sizeof first);
        for(int i=1;i<=n;i++)
        {
            scanf("%d%d%d",&u,&v,&w);
            S=min(S,u-1),T=max(T,v);
            addedge(u-1,v,w);
        }
        for(int i=S;i<T;i++)
        {
            addedge(i,i+1,0);
            addedge(i+1,i,-1);
        }
        SPFA(S);
        printf("%d\n",dis[T]);
    }
    return 0;
}
```

4.2 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
const int INF = 0x3f3f3f3f;
const int maxn = 205;
const int maxm=1005;
int n, m, s, t, u, v, w,dis[maxn];
typedef pair<int, int> pi;
struct Edge{int v,w,nxt;}e[maxm<<1];
int head[maxn],tot;
void addedge(int u, int v, int w){
    e[tot].v=v; e[tot].w=w; e[tot].nxt=head[u]; head[u]=tot++;
}
void dijkstra(int S){
    priority_queue<pi, vector<pi>, greater<pi> > q;
    for (int i = 1; i <= n; i++) dis[i] = INF;
    dis[S] = 0; q.push(make_pair(0, S));
    while (!q.empty()){
        pi p = q.top(); q.pop();
        if (dis[p.second] != p.first) continue;
        for (int i = head[p.second]; i; i=e[i].nxt){
            int v = e[i].v, w=e[i].w;
            if (dis[v] > dis[p.second] + w){
                dis[v] = dis[p.second] + w;
                q.push(make_pair(dis[v], v));
            }
        }
    }
}

int main()
{
    while (~scanf("%d%d", &n, &m)){
        for (int i = 1; i <= n; i++) head[i]=0; tot=1;
        for (int i = 1; i <= m; i++){
            scanf("%d %d %d",&u,&v,&w);
            u++; v++;
            addedge(u, v, w);
            addedge(v, u, w);
        }
        scanf("%d %d",&s,&t);
        s++; t++;
        dijkstra(s);
        if (dis[t] == INF) printf("-1\n");
        else printf("%d\n",dis[t]);
    }
    return 0;
}
```

4.3 欧拉回路

```
#include<bits/stdc++.h>
using namespace std;
```

```
typedef long long ll;
const int maxn=1e5+5;
int t,n,m; //t=1: 无向图 t=2: 有向图

struct Edge{int v,nxt;}e[maxn<<2];
int head[maxn],in[maxn],out[maxn],tot=1;
bool vis[maxn<<1];
void addedge(int u,int v){
   e[++tot].v=v;e[tot].nxt=head[u];head[u]=tot;
}
int edge[maxn<<1],cnt;
void dfs(int u){
   for(int &i=head[u];i;i=e[i].nxt){
      int v=e[i].v,j=i;
      if(vis[j>>1])continue;
      vis[j>>1]=true;
      dfs(v);
      edge[++cnt]=j;
   }
}

int main()
{
   scanf("%d",&t);
   scanf("%d %d",&n,&m);
   int u,v;
   for(int i=1;i<=m;i++){
      scanf("%d %d",&u,&v);
      addedge(u,v);
      if(t==1)addedge(v,u), in[u]++, out[v]++;
      else tot++, in[v]++, out[u]++;
   }

   if(t==1){
      for(int i=1;i<=n;i++)if((in[i]+out[i])&1){
         puts("NO");
         return 0;
      }
   }
   else{
      for(int i=1;i<=n;i++)if(in[i]!=out[i]){
         puts("NO");
         return 0;
      }
   }

   dfs(u);
   if(cnt!=m)puts("NO");
   else{
      puts("YES");
      while(cnt)printf("%d ",(edge[cnt]&1)?-(edge[cnt]>>1):(edge[cnt]>>1)),cnt--;
   }
   return 0;
}
```

```
//HierHolzer Algorithm, O(|E|+|V|)
void printCircuit(vector< vector<int> > adj)
{
   // adj represents the adjacency list of the directed graph
   // edge_count represents the number of edges emerging from a vertex
   unordered_map<int,int> edge_count;

   for (int i=0; i<adj.size(); i++)
   {
      //find the count of edges to keep track
```

```
        //of unused edges
        edge_count[i] = adj[i].size();
    }

    if (!adj.size())
    return; //empty graph

    // Maintain a stack to keep vertices
    stack<int> curr_path;

    // vector to store final circuit
    vector<int> circuit;

    // start from any vertex
    curr_path.push(0);
    int curr_v = 0; // Current vertex

    while (!curr_path.empty())
    {
        // If there's remaining edge
        if (edge_count[curr_v])
        {
            // Push the vertex
            curr_path.push(curr_v);

            // Find the next vertex using an edge
            int next_v = adj[curr_v].back();

            // and remove that edge
            edge_count[curr_v]--;
            adj[curr_v].pop_back();

            // Move to next vertex
            curr_v = next_v;
        }

        // back-track to find remaining circuit
        else
        {
            circuit.push_back(curr_v);

            // Back-tracking
            curr_v = curr_path.top();
            curr_path.pop();
        }
    }

    // we've got the circuit, now print it in reverse
    for (int i=circuit.size()-1; i>=0; i--)
    {
        cout << circuit[i];
        if (i)
        cout<<" -> ";
    }
}
```

4.4 K 短路

4.4.1 可持久化可并堆 1

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn=5005;
const int maxm=2e5 + 5;
const double eps=1e-8;

int n, m, vis[maxn];
double E, ei, dis[maxn];
//Graph
struct Edge {
   int to, next;
   double w;
} e[maxm << 1];
int head[maxn], cnt, fa[maxn], cov[maxm << 1];
inline void graph_init() {
   cnt=0;
   memset(head, -1, sizeof(head));
}
inline void addedge(int u, int v, double w) {
   e[cnt] = (Edge) {v, head[u], w}, head[u] = cnt++;
   e[cnt] = (Edge) {u, head[v], w}, head[v] = cnt++;
}


//Persistent Heap
struct Heap_node {
   int ls, rs, dis, ed;
   double w;
} tr[maxm * 20];
int tot,rt[maxn];
inline int newnode(double w, int ed) {
   int x = ++tot;
   tr[x].w = w, tr[x].dis = 1, tr[x].ed = ed;
   return x;
}
int Merge(int x, int y) {
   if (!x || !y) return x + y;
   if (tr[x].w - tr[y].w >= eps) swap(x, y);
   int p = ++tot;
   tr[p] = tr[x], tr[p].rs = Merge(tr[p].rs, y);
   if (tr[tr[p].ls].dis < tr[tr[p].rs].dis) swap(tr[p].ls, tr[p].rs);
   tr[p].dis = tr[tr[x].rs].dis + 1;
   return p;
}

//dijkstra
struct node{
   double w;
   int id;
   node(){}
   node(double ww,int ID): w(ww), id(ID){}
   bool operator <(const node &b) const{return w > b.w;}
};
priority_queue < node > q;
inline void dijkstra(int S) {
   for(int i=0;i<=n;i++)dis[i]=DBL_MAX;
   for (int i = 1; i <= n; ++i) vis[i] = 0;
   dis[S] = 0, q.push(node(0,S));
   while (!q.empty()) {
      int u = q.top().id; q.pop();
      if (vis[u]) continue;
      vis[u] = 1;
      for (int i = head[u]; ~i; i = e[i].next)
      if (i & 1) {
         int v = e[i].to;
         if (dis[v] - (dis[u] + e[i].w) >= eps) {
            dis[v] = dis[u] + e[i].w;
            q.push(node(dis[v], v));
            }
        }
    }
}

void dfs(int u) {
    vis[u] = 1;
    for (int i = head[u]; ~i; i = e[i].next)
    if (i & 1) {
        double w = e[i].w;
        int v=e[i].to;
        if (fabs(dis[u] + w - dis[v]) < eps && !vis[v])
        fa[v] = u, cov[i ^ 1] = 1, dfs(v);
    }
}

int main() {
    scanf("%d %d %lf", &n, &m, &E);
    graph_init();
    int u, v;
    for (int i = 1; i <= m; ++i) {
        scanf("%d %d %lf", &u, &v, &ei);
        addedge(u, v, ei);
    }
    dijkstra(n);
    for (int i = 1; i <= n; ++i) vis[i] = 0;
    dfs(n);
    for (int i = 0; i < cnt; i += 2)
    if (!cov[i]) {
        u = e[i ^ 1].to, v = e[i].to;
        if (dis[u] == dis[0] || dis[v] == dis[0]) continue;
        rt[u] = Merge(rt[u], newnode(dis[v] + e[i].w - dis[u], v));
    }
    for (int i = 1; i <= n; ++i) q.push(node(dis[i], i));
    for (int i = 1; i <= n; ++i) {
        u = q.top().id; q.pop();
        if (fa[u]) rt[u] = Merge(rt[u], rt[fa[u]]);
    }
    int ans=0;
    if (dis[1] - E < eps) E -= dis[1], ++ans;
    if (rt[1]) q.push(node(tr[rt[1]].w, rt[1]));
    while (!q.empty()) {
        u = q.top().id;
        double cur = q.top().w, W = dis[1] + cur;
        if (W - E >= eps) break;
        q.pop(), E -= W, ++ans;
        for (int i = 0; i < 2; ++i) {
            v = i ? tr[u].rs : tr[u].ls;
            if (v) q.push(node(tr[v].w - tr[u].w + cur, v));
        }
        if (rt[tr[u].ed]) q.push(node(tr[rt[tr[u].ed]].w + cur, rt[tr[u].ed]));
    }
    printf("%d\n", ans);
    return 0;
}
```

4.4.2 可持久化可并堆 2

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn=1005;
const int maxm=1e5 + 5;
```

```
int n, m, s, t, k, vis[maxn], dis[maxn];
//Graph
struct Edge {
    int to, next, w;
} e[maxm << 1];
int head[maxn], cnt, fa[maxn], cov[maxm << 1];
inline void graph_init() {
    cnt=0;
    memset(head, -1, sizeof(head));
}
inline void addedge(int u, int v, int w) {
    e[cnt] = (Edge) {v, head[u], w}, head[u] = cnt++;
    e[cnt] = (Edge) {u, head[v], w}, head[v] = cnt++;
}


//Persistent Heap
struct Heap_node {
    int ls, rs, dis, ed, w;
} tr[maxm * 20];
int tot,rt[maxn];
inline int newnode(int w, int ed) {
    int x = ++tot;
    tr[x].w = w, tr[x].dis = 1, tr[x].ed = ed;
    return x;
}
int Merge(int x, int y) {
    if (!x || !y) return x + y;
    if (tr[x].w >= tr[y].w) swap(x, y);
    int p = ++tot;
    tr[p] = tr[x], tr[p].rs = Merge(tr[p].rs, y);
    if (tr[tr[p].ls].dis < tr[tr[p].rs].dis) swap(tr[p].ls, tr[p].rs);
    tr[p].dis = tr[tr[x].rs].dis + 1;
    return p;
}

//dijkstra
struct node{
    int w, id;
    node(){}
    node(int ww,int ID): w(ww), id(ID){}
    bool operator <(const node &b) const{return w > b.w;}
};
priority_queue < node > q;
inline void dijkstra(int S) {
    for(int i=0;i<=n;i++)dis[i]=1e9+7;
    for (int i = 1; i <= n; ++i) vis[i] = 0;
    dis[S] = 0, q.push(node(0,S));
    while (!q.empty()) {
        int u = q.top().id; q.pop();
        if (vis[u]) continue;
        vis[u] = 1;
        for (int i = head[u]; ~i; i = e[i].next)
        if (i & 1) {
            int v = e[i].to;
            if (dis[v] > dis[u] + e[i].w) {
                dis[v] = dis[u] + e[i].w;
                q.push(node(dis[v], v));
            }
        }
    }
}

void dfs(int u) {
    vis[u] = 1;
    for (int i = head[u]; ~i; i = e[i].next)
    if (i & 1) {
        int w = e[i].w;
        int v=e[i].to;
        if (dis[u] + w == dis[v] && !vis[v])
        fa[v] = u, cov[i ^ 1] = 1, dfs(v);
    }
}

int main() {
    scanf("%d %d", &n, &m);
    graph_init();
    int u, v, w;
    for (int i = 1; i <= m; ++i) {
        scanf("%d %d %d", &u, &v, &w);
        addedge(u, v, w);
    }
    scanf("%d %d %d",&s,&t,&k);k+=(s==t);
    dijkstra(t);

    if(dis[s]==dis[0]){
        printf("-1\n");
        return 0;
    }

    for (int i = 1; i <= n; ++i) vis[i] = 0;
    dfs(t);
    for (int i = 0; i < cnt; i += 2)
    if (!cov[i]) {
        u = e[i ^ 1].to, v = e[i].to;
        if (dis[u] == dis[0] || dis[v] == dis[0]) continue;
        rt[u] = Merge(rt[u], newnode(dis[v] + e[i].w - dis[u], v));
    }
    for (int i = 1; i <= n; ++i) q.push(node(dis[i], i));
    for (int i = 1; i <= n; ++i) {
        u = q.top().id; q.pop();
        if (fa[u]) rt[u] = Merge(rt[u], rt[fa[u]]);
    }
    int cnt=1,ans=-1;
    if (rt[s]) q.push(node(tr[rt[s]].w, rt[s]));
    while (!q.empty()) {
        u = q.top().id;
        int cur = q.top().w;
        ans = dis[s] + cur;
        q.pop(), ++cnt;
        if(cnt == k)break;
        for (int i = 0; i < 2; ++i) {
            v = i ? tr[u].rs : tr[u].ls;
            if (v) q.push(node(tr[v].w - tr[u].w + cur, v));
        }
        if (rt[tr[u].ed]) q.push(node(tr[rt[tr[u].ed]].w + cur, rt[tr[u].ed]));
    }
    printf("%d\n", ans);
    return 0;
}
```

4.5 差分约束

$$
\begin{array}{cccc}
 & x_a - x_b \geq c & x_b - x_a \leq -c & add(a,b,-c) \\
 & x_a - x_b \leq c & x_a - x_b \leq c & add(b,a,c) \\
x_a = x_b & x_a - x_b \leq 0, x_b - x_a \leq 0 & & add(a,b,0), add(b,a,0)
\end{array}
$$

if exist negative circle,the answer is no,otherwise the answer is yes

4.6 LCA

4.6.1 st 表/倍增

```
int dep[40005],dis[40005],par[40005][25];
void dfs(int u,int fa,int d,int dd){
    dep[u] = d;//深度
    dis[u] = dd;//距离
    if(u==1){
        for(int i = 0;i<20;i++)par[u][i] = 1;
    }
    else{
        par[u][0] = fa;
        for(int i = 1;i<20;i++){
            par[u][i] = par[par[u][i-1]][i-1];
        }
    }
    for(int i=0;i<G[u].size();i++){
        int v = G[u][i].fi,c = G[u][i].se;
        if(v==fa)continue;
        dfs(v,u,d+1,dd+c);
    }
}
int Jump(int u,int d){
    for(int j = 19;j>=0;j--){
        if((1<<j)&d){
            u = par[u][j];
        }
    }
    return u;
}
int lca(int u,int v){
    if(dep[u]<dep[v])swap(u,v);
    u = Jump(u,dep[u]-dep[v]);
    if(u==v)return u;
    for(int i = 19;i>=0;i--){
        if(par[u][i]!=par[v][i]){
            u = par[u][i];
            v = par[v][i];
        }
    }
    return par[u][0];
}
```

4.6.2 tarjan/dfs

```
#include<cstdio>
#define N 420000
struct hehe{
    int next;
    int to;
    int lca;
};
hehe edge[N];//树的链表
hehe qedge[N];//需要查询LCA的两节点的链表
int n,m,p,x,y;
int num_edge,num_qedge,head[N],qhead[N];
int father[N];
int visit[N];//判断是否被找过
void add_edge(int from,int to){//建立树的链表
    edge[++num_edge].next=head[from];
    edge[num_edge].to=to;
    head[from]=num_edge;
}
void add_qedge(int from,int to){//建立需要查询LCA的两节点的链表
    qedge[++num_qedge].next=qhead[from];
    qedge[num_qedge].to=to;
    qhead[from]=num_qedge;
}
int find(int z){//找爹函数
    if(father[z]!=z)
    father[z]=find(father[z]);
    return father[z];
}
int dfs(int x){//把整棵树的一部分看作以节点x为根节点的小树
    father[x]=x;//由于节点x被看作是根节点，所以把x的father设为它自己
    visit[x]=1;//标记为已被搜索过
    for(int k=head[x];k;k=edge[k].next)//遍历所有与x相连的节点
    if(!visit[edge[k].to]){//若未被搜索
        dfs(edge[k].to);//以该节点为根节点搞小树
        father[edge[k].to]=x;//把x的孩子节点的father重新设为x
    }
    for(int k=qhead[x];k;k=qedge[k].next)//搜索包含节点x的所有询问
    if(visit[qedge[k].to]){//如果另一节点已被搜索过
        qedge[k].lca=find(qedge[k].to);//把另一节点的祖先设为这两个节点的最近公共祖先
        if(k%2)//由于将每一组查询变为两组，所以2n-1和2n的结果是一样的
        qedge[k+1].lca=qedge[k].lca;
        else
        qedge[k-1].lca=qedge[k].lca;
    }
}
int main(){
    scanf("%d%d%d",&n,&m,&p);//输入节点数，查询数和根节点
    for(int i=1;i<n;++i){
        scanf("%d%d",&x,&y);//输入每条边
        add_edge(x,y);
        add_edge(y,x);
    }
    for(int i=1;i<=m;++i){
        scanf("%d%d",&x,&y);//输入每次查询，考虑(u,v)时若查找到u但v未被查找，所以将(u,v)(v,u)全部记录
        add_qedge(x,y);
        add_qedge(y,x);
    }
    dfs(p);//进入以p为根节点的树的深搜
    for(int i=1;i<=m;i++)
    printf("%d ",qedge[i*2].lca);//两者结果一样，只输出一组即可
    return 0;
}
```

4.7 最小生成树

4.7.1 Prim

```
mark[0]=0;
for(int i=1;i<n;i++)lowcost[i]=dis[0][i];
lowcost[0]=0;
int a=0;
```

```
for(int j=1;j<n;j++)
{
    double mi=1e12;
    for(int i=0;i<n;i++)
    {
        if(mark[i]==0 && lowcost[i]<mi)
        {
        mi=lowcost[i];
        a=i;
        }
    }
    mark[a]=1;
    for(int i=0;i<n;i++)
    {
        if(mark[i]==0 && lowcost[i]>dis[a][i])
        lowcost[i]=dis[a][i];
    }
}
```

4.7.2 Kruskal

```
int merge(int x,int y)
{
    int fx=find(x),fy=find(y);
    if(fx!=fy)
    {
        fa[fx]=fy;
        return 1;
    }
    return 0;
}


sort(e,e+k);
for(int i=0;i<k;i++)
{
    if(merge(e[i].from,e[i].to)) //并查集
    {
        cost+=e[i].val;
        merge(e[i].from,e[i].to);
        if(++num==p-1)break;
    }
}
```

4.8 曼哈顿距离最小生成树

```
//求连接两点代价为两点间曼哈顿距离的平面点最小生成树
//对于每个点最多连8条边(每45°的范围内一条)
//然后Kruskal，复杂度O(nlogn)
#include<bits/stdc++.h>
using namespace std;
const int MAXN=100010;
const int MAXE=MAXN*4;
const int INF=0x3f3f3f3f;
/*kruskal alrorithm*/
int father[MAXN],n;
int parent(int u)
{
    while(father[u]!=u)
    {
        father[u]=father[father[u]];
        u=father[u];
    }
    return u;
}
bool connect(int u,int v)
{
    int fu=parent(u);
    int fv=parent(v);
    if(fu==fv) return false;
    father[fu]=fv;
    return true;
}
struct Point
{
    int x,y,id;//id is for union_found
    bool operator <(const Point &p) const
    {
        if(x==p.x) return y<p.y;
        return x<p.x;
    }
};
Point point[MAXN];
/*数状数组*/
struct Node
{
    int len,id;//len is to find min(x+y), id is for union_found
    void init(){len=INF;id=-1;}
};
Node c[MAXN<<2];
/*倒叙的树状数组，求的是后缀*/
void add(int x,Point &p)//push point into tree
{
    int len=p.x+p.y;
    while(x>0)
    {
        if(c[x].len>len)
        {
            c[x].id=p.id;
            c[x].len=len;
        }
        x-=x&(-x);
    }
}
/*在区间[x,y]求最小值*/
Node query(int x,int y)//从大于point[i].y-point[i].x的节点找最小值
{
    Node t;t.init();
    while(x<=y)
    {
        if(t.len>c[x].len)
        {
            t=c[x];
        }
        x+=x&(-x);
    }
    return t;
}
struct Edge
{
    int u,v,w;
    Edge(){}
    Edge(int u,int v,int w):u(u),v(v),w(w){}
    bool operator <(const Edge &ee) const
    {
        return w<ee.w;
    }
};
Edge edge[MAXE];
```

```
int cnt;
void addEdge(int u,int v,int val)
{
   edge[cnt++]=Edge(u,v,val);
}
int cpy[MAXN],arr[MAXN];
void solve(int n)
{
   sort(point+1,point+1+n);
   for(int i=1;i<=n;i++)
   {
      arr[i]=cpy[i]=point[i].y-point[i].x;
   }
   sort(cpy+1,cpy+1+n);
   int cc=unique(cpy+1,cpy+1+n)-cpy;
   for(int i=1;i<=n;i++)
   {
      arr[i]=lower_bound(cpy+1,cpy+cc,arr[i])-cpy;
   }
   for(int i=1;i<=cc;i++) c[i].init();
   for(int i=n;i>0;i--)
   {
      Node t=query(arr[i],cc);
      if(t.id!=-1) addEdge(point[i].id,t.id,abs(point[i].x+point[i].y-t.len));
      add(arr[i],point[i]);
   }
}
long long kruskal_mst()
{
   int u,v;
   long long sum=0;
   sort(edge,edge+cnt);
   for(int i=0;i<cnt;i++)
   {
      u=edge[i].u;v=edge[i].v;
      if(connect(u,v))
      {
         sum+=(long long) edge[i].w;
      }
   }
   return sum;
}
int main()
{
   int cas=1;
   while(scanf("%d",&n)!=EOF,n)
   {
      for(int i=1;i<=n;i++)
      {
         scanf("%d%d",&point[i].x,&point[i].y);
         point[i].id=i;
      }
      for(int i=1;i<=n;i++)
      {
         father[i]=i;
      }
      cnt=0;
      solve(n);
      for(int i=1;i<=n;i++)
      point[i].y=-point[i].y;
      solve(n);
      for(int i=1;i<=n;i++)
      point[i].y=-point[i].y,swap(point[i].x,point[i].y);
      solve(n);
      for(int i=1;i<=n;i++)
```

```
        point[i].y=-point[i].y;
        solve(n);
        printf("Case %d: Total Weight = %lld\n",cas++,kruskal_mst());
    }
}
```

4.9 斯坦纳树

transfer from its subset:$f[i][sta] = \min_{s \in sta} \{f[i][s] + f[i][C_{sta}s] - val[i]\}$

transfer from extend nodes:take node i as the new node,$f[i][j] = \min\{f[k][j] + val[i]\}$

use spfa algorithm to maintain this transfer

```
//求让给定点集联通的最小代价, 可以通过增加额外点使代价减少
//子集状压dp
//枚举sta的子集:for(int s = sta; s; s = (s - 1) & sta)
#include<bits/stdc++.h>
using namespace std;
const int limit = 1050;
const int INF = 1e9;
inline int read() {
    char c = getchar(); int x = 0, f = 1;
    while(c < '0' || c > '9') {if(c == '-') f = -1; c = getchar();}
    while(c >= '0' && c <= '9') {x = x * 10 + c - '0'; c = getchar();}
    return x * f;
}
#define MP(i,j) make_pair(i,j)
#define se second
#define fi first
#define Pair pair<int,int>
int N, M, tot = 0;
int a[12][12], f[12][12][limit];
int xx[5] = {-1, +1, 0, 0};
int yy[5] = {0, 0, -1, +1};
int vis[12][12];
struct PRE {
    int x, y, S;
}Pre[12][12][limit];
queue<Pair>q;
void SPFA(int cur) {
    while(q.size() != 0) {
        Pair p = q.front();q.pop();
        vis[p.fi][p.se] = 0;
        for(int i = 0; i <4; i++) {
            int wx = p.fi + xx[i], wy = p.se + yy[i];
            if(wx < 1 || wx > N || wy < 1 || wy > M) continue;
            if(f[wx][wy][cur] > f[p.fi][p.se][cur] + a[wx][wy]) {
                f[wx][wy][cur] = f[p.fi][p.se][cur] + a[wx][wy];
                Pre[wx][wy][cur] = (PRE){p.fi, p.se, cur};
                if(!vis[wx][wy])
                vis[wx][wy] = 1, q.push(MP(wx,wy));
            }
        }
    }
}
void dfs(int x, int y, int now) {
    vis[x][y] = 1;
    PRE tmp = Pre[x][y][now];
    if(tmp.x == 0 && tmp.y == 0) return;
    dfs(tmp.x, tmp.y, tmp.S);
    if(tmp.x == x && tmp.y == y) dfs(tmp.x, tmp.y, now - tmp.S);
}
int main() {
   //freopen("a.in", "r", stdin);
   N = read(); M = read();
   memset(f, 0x3f, sizeof(f));
   for(int i = 1; i <= N; i++)
   for(int j = 1; j <= M; j++) {
      a[i][j] = read();
      if(a[i][j] == 0)
      f[i][j][1 << tot] = 0, tot++;
   }
   int limit = (1 << tot) - 1;
   for(int sta = 0; sta <= limit; sta++) {
      for(int i = 1; i<= N; i++)
      for(int j = 1; j <= M;j++) {
         for(int s = sta; s; s = (s - 1) & sta) {
            if(f[i][j][s] + f[i][j][sta - s] - a[i][j] < f[i][j][sta])
            f[i][j][sta] = f[i][j][s] + f[i][j][sta - s] - a[i][j],
            Pre[i][j][sta] = (PRE){i,j,s};
         }
         if(f[i][j][sta] < INF) q.push(MP(i,j)), vis[i][j] = 1;
      }
      SPFA(sta);
   }
   int ansx, ansy, flag = 0;
   for(int i = 1; i <= N && !flag; i++)
   for(int j = 1; j <= M; j++)
   if(!a[i][j]) {ansx = i, ansy = j; flag = 1; break;}
   printf("%d\n",f[ansx][ansy][limit]);
   memset(vis, 0, sizeof(vis));
   dfs(ansx, ansy, limit);
   for(int i = 1; i <= N; i++, puts("")) {
      for(int j = 1; j <= M; j++) {
         if(a[i][j] == 0) putchar('x');
         else if(vis[i][j]) putchar('o');
         else putchar('_');
      }
   }
   return 0;
}
```

4.10 最小树形图 Edmonds 算法

```
//也称朱刘算法，O(nm)
//1.对于每个点，选择它入度最小的那条边
//2.如果没有环，算法终止；否则进行缩环并更新其他点到环的距离。
bool solve() {
   ans = 0;
   int u, v, root = 0;
   for (;;) {
      f(i, 0, n) in[i] = 1e100;
      f(i, 0, m) {
         u = e[i].s;
         v = e[i].t;
         if (u != v && e[i].w < in[v]) {
            in[v] = e[i].w;
            pre[v] = u;
         }
      }
      f(i, 0, m) if (i != root && in[i] > 1e50) return 0;
      int tn = 0;
      memset(id, -1, sizeof id);
      memset(vis, -1, sizeof vis);
      in[root] = 0;
      f(i, 0, n) {
            ans += in[i];
            v = i;
            while (vis[v] != i && id[v] == -1 && v != root) {
                vis[v] = i;
                v = pre[v];
            }
            if (v != root && id[v] == -1) {
                for (int u = pre[v]; u != v; u = pre[u]) id[u] = tn;
                id[v] = tn++;
            }
        }
        if (tn == 0) break;
        f(i, 0, n) if (id[i] == -1) id[i] = tn++;
        f(i, 0, m) {
            u = e[i].s;
            v = e[i].t;
            e[i].s = id[u];
            e[i].t = id[v];
            if (e[i].s != e[i].t) e[i].w -= in[v];
        }
        n = tn;
        root = id[root];
    }
    return ans;
}
```

4.11 二分图

4.11.1 定理

```
/*
最大匹配数：最大匹配的匹配边的数目
最小点覆盖数：选取最少的点，使任意一条边至少有一个端点被选择
最大独立数：选取最多的点，使任意所选两点均不相连
最小路径覆盖数：对于一个 DAG（有向无环图），选取最少条简单路径，使得每个顶点属于且仅属于一条路径。路径长可以为 0（即单个
    点）。

1.最大匹配数=最小点覆盖数
2.最大独立数=顶点数-最大匹配数
3.最小路径覆盖数=顶点数-原DAG图的拆点二分图的最大匹配数
*/
```

4.11.2 匈牙利算法

```
//无权最大匹配
#include<bits/stdc++.h>
#define pb push_back
#define fi first
#define se second
using namespace std;
typedef long long ll;
const int maxn=233;
int n,m;
int c[maxn],vis[maxn],pre[maxn],flag;
vector<int>G[maxn];
void judge(int u,int f)
{
    if(flag)return;
    c[u]=f;
    for(int i:G[u]){
        if(c[i]){
            if(c[i]==c[u]){
                flag=1;
                break;
            }
            continue;
        }
        judge(i,3-f);
    }
}
bool find(int u)
{
    for(int i:G[u]){
        if(vis[i]==0){
            vis[i]=1;
            if(pre[i] == 0 || find(pre[i])){
                pre[i]=u;
                return true;
            }
        }
    }
    return false;
}
int main()
{
    while(~scanf("%d %d",&n,&m)){
        memset(c,0,sizeof(c));
        memset(pre,0,sizeof(pre));
        flag=0;
        for(int i=1;i<=n;i++)G[i].clear();
        int u,v;
        for(int i=1;i<=m;i++){
            scanf("%d %d",&u,&v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        judge(1,1);
        if(flag){
            printf("No\n");
            continue;
        }
        int ans=0;
        for(int i=1;i<=n;i++){
            memset(vis,0,sizeof(vis));
            if(find(i)) ans++;
        }
        printf("%d\n",ans/2);
    }
    return 0;
}
```

4.11.3 KM

```
//带权最大匹配
#include<bits/stdc++.h>
#define pb push_back
#define fi first
#define se second
using namespace std;
typedef long long ll;
const int maxn=369;
const int INF=0x3f3f3f3f;
int n,nx,ny;
int lx[maxn],ly[maxn],slack[maxn],match[maxn];
bool visx[maxn],visy[maxn];
int G[maxn][maxn];
bool findpath(int x)
{
    int tempDelta;
    visx[x] = true;
    for(int y = 0 ; y < ny ; ++y){
        if(visy[y]) continue;
        tempDelta = lx[x] + ly[y] - G[x][y];
        if(tempDelta == 0){//(x,y)在相等子图中
            visy[y] = true;
            if(match[y] == -1 || findpath(match[y])){
                match[y] = x;
                return true;
            }
        }
        else if(slack[y] > tempDelta)
            slack[y] = tempDelta;
        //(x,y)不在相等子图中且y不在交错树中
    }
    return false;
}
int KM()
{
    memset(match,-1,sizeof(match));
    memset(ly,0,sizeof(ly));
    for(int i=0;i<nx;i++){
        lx[i]=-INF;
        for(int j=0;j<ny;j++)
            if(G[i][j]>lx[i])lx[i]=G[i][j];
    }

    for(int x = 0 ; x < nx ; ++x) {
        for(int y = 0 ; y < ny ; ++y) slack[y] = INF;
        while(true) {
            memset(visx,false,sizeof(visx));
            memset(visy,false,sizeof(visy));
            if(findpath(x)) break;
            else {
                int delta = INF;
                for(int j = 0 ; j < ny ; ++j)
                if(!visy[j] && delta > slack[j])
                delta = slack[j];
                for(int i = 0 ; i < nx ; ++i)
                if(visx[i]) lx[i] -= delta;
                for(int j = 0 ; j < ny ; ++j){
                    if(visy[j]) ly[j] += delta;
                    else slack[j] -= delta;
                }
            }
        }
    }
    int ans=0;
    for(int i=0;i<ny;i++)if(match[i]!=-1)ans+=G[match[i]][i];
    return ans;
}
int main()
{
    while(~scanf("%d",&n)){
        nx=ny=n;
        memset(G,0,sizeof(G));
        for(int i=0;i<n;++i){
            for(int j=0;j<n;++j){
                scanf("%d",&G[i][j]);
            }
        }
        printf("%d\n",KM());
    }
    return 0;
}
```

4.12 最大流

4.12.1 Dinic

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 60009; //X 集合中的顶点数上限
const int MAXM = 50009<<6; // 总的边数上限
const int INF = 0x3f3f3f3f;
int head[MAXN],cur[MAXN],tot; //cur[]:当前弧优化
int S,T; // S 是源点, T 是汇点
int d[MAXN]; // 存储每个顶点的层次

struct Edge{
    int v,c,nxt;
    // v 是指边的另一个顶点, c 表示容量
}e[MAXM];

void init(){
    memset(head,-1,sizeof(head));
    tot=0;
}

void addedge(int u,int v,int c){
    // 插入一条从 u 连向 v, 容量为 c 的弧
    e[tot].v=v;e[tot].c=c;
    e[tot].nxt=head[u];
    head[u]=tot++;

    e[tot].v=u;e[tot].c=0;
    e[tot].nxt=head[v];
    head[v]=tot++;
}

bool bfs(){
    // bfs构建层次图G_L
    memset(d,-1,sizeof(d));
    queue<int> q;
    q.push(S);
    d[S]=0;
    while(!q.empty()){
        int u=q.front();
        q.pop();
        for(int i=head[u];i!=-1;i=e[i].nxt){
            int v=e[i].v;
            if(e[i].c>0&&d[v]==-1){
                q.push(v);
                d[v]=d[u]+1;
            }
        }
    }
    return (d[T]!=-1);
}

int dfs(int u,int flow){
    // dfs在层次图G_L中寻找增广路径
    // flow 表示当前搜索分支的流量上限
    if(u==T){
        return flow;
    }
    int res=0;
    for(int& i=cur[u];i!=-1;i=e[i].nxt){
        int v=e[i].v;
        if(e[i].c>0&&d[u]+1==d[v]){
            int tmp=dfs(v,min(flow,e[i].c));
            // 递归计算顶点 v, 用 c(u, v) 来更新当前流量上限
            flow-=tmp;
            e[i].c-=tmp;
            res+=tmp;
            e[i^1].c+=tmp; // 修改反向弧的容量
            if(flow==0){ // 流量达到上限，不必继续搜索了
                break;
            }
        }
    }
    if(res==0){
        // 当前没有经过顶点 u 的可行流，不再搜索顶点 u
        d[u]=-1;
    }
    return res;
}

int maxflow(){ // 函数返回值就是最大流的结果
    int res=0;
    while(bfs()){
        memcpy(cur,head,sizeof(cur));
        res+=dfs(S,INF); // 初始流量上限为 INF
    }
    return res;
}

int main(){
    int m,n;
    while(~scanf("%d %d",&n,&m)){
    init();
    S=0;T=n+m+1;
    int c;
    for(int i=1;i<=n;i++){
        scanf("%d",&c);
        addedge(S,i,c);
    }
    int u,v,sum=0;

    for(int i=1;i<=m;i++){
        scanf("%d %d %d",&u,&v,&c);
        addedge(u,n+i,INF);
        addedge(v,n+i,INF);
        addedge(n+i,T,c);
        sum+=c;
    }
    printf("%d\n",sum-maxflow());
}
return 0;
}
```

4.12.2 HLPP

```
//最高标号预流推进算法(High Level Preflow Push), 复杂度O(sqrt(m)*n^2)
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> pi;
const int maxn=6e4+5;
const int maxm=maxn<<3;
const int INF=0x3f3f3f3f;

int n,m,V,S,T;
struct edge {
    int nxt, v, c;
} e[maxm];
int head[maxn], tot = 1;
```

```
void add_path(int u, int v, int c) {
   e[++tot] = (edge) {head[u], v, c}, head[u] = tot;
}
void add_flow(int u, int v, int c) {
   add_path(u, v, c);
   add_path(v, u, 0);
}

int ht[maxn], ex[maxn], gap[maxn]; // 高度; 超额流; gap 优化
bool bfs_init() {
   memset(ht, 0x3f, sizeof(ht));
   queue<int> q;
   q.push(T), ht[T] = 0;
   while (!q.empty()) { // 反向 BFS, 遇到没有访问过的结点就入队
      int u = q.front(); q.pop();
      for (int i = head[u]; i; i = e[i].nxt) {
         int v = e[i].v;
         if (e[i ^ 1].c && ht[v] > ht[u] + 1) ht[v] = ht[u] + 1, q.push(v);
      }
   }
   return ht[S] != INF; // 如果图不连通, 返回 0
}
struct cmp {
   bool operator()(int a, int b) const {
      return ht[a] < ht[b];
   }
}; // 伪装排序函数
priority_queue<int, vector<int>, cmp> pq; // 将需要推送的结点以高度高的优先
bool vis[maxn]; // 是否在优先队列中
int push(int u) { // 尽可能通过能够推送的边推送超额流
   for (int i = head[u]; i; i = e[i].nxt) {
      int v = e[i].v, c = e[i].c;
      if (!c || ht[u] != ht[v] + 1) continue;
      int k = min(c, ex[u]); // 取到剩余容量和超额流的最小值
      ex[u] -= k, ex[v] += k, e[i].c -= k, e[i ^ 1].c += k; // push
      if (v != S && v != T && !vis[v])
      pq.push(v), vis[v] = 1; // 推送之后, v 必然溢出, 则入堆, 等待被推送
      if (!ex[u]) return 0; // 如果已经推送完就返回
   }
   return 1;
}
void relabel(int u) { // 重贴标签 (高度)
   ht[u] = INF;
   for (int i = head[u]; i; i = e[i].nxt)
   if (e[i].c) ht[u] = min(ht[u], ht[e[i].v]);
   ++ht[u];
}
int hlpp() { // 返回最大流
   if (!bfs_init()) return 0; // 图不连通
   ht[S] = V;
   memset(gap, 0, sizeof(gap));
   for (int i = 1; i <= V; i++)
   if (ht[i] != INF) gap[ht[i]]++; // 初始化 gap
   for (int i = head[S]; i; i = e[i].nxt) {
      int v = e[i].v, c = e[i].c; // 队列初始化
      if (!c) continue;
      ex[S] -= c, ex[v] += c, e[i].c -= c, e[i ^ 1].c += c; // 注意取消 w 的引用
      if (v != S && v != T && !vis[v]) pq.push(v), vis[v] = 1; // 入队
   }
   while (!pq.empty()) {
      int u = pq.top(); pq.pop(); vis[u] = 0;
      while (push(u)) { // 仍然溢出
         // 如果 u 结点原来所在的高度没有结点了, 相当于出现断层
         if (!--gap[ht[u]])
```

```
            for (int i = 1; i <= V; i++)
            if (i != S && i != T && ht[i] > ht[u] && ht[i] < V + 1) ht[i] = V + 1;
            relabel(u);
            ++gap[ht[u]]; // 新的高度，更新 gap
        }
    }
    return ex[T];
}
void init(int N){
    V=N; tot=1;
    for(int i=1;i<=V;i++)head[i]=ex[i]=0;
}

int p[maxn];
void solve(){
    S=n+m+1; T=n+m+2;
    init(T);
    int u,v,c,sum=0;
    for(int i=1;i<=n;i++){
        read(c);
        add_flow(S,i,c);
    }
    for(int i=1;i<=m;i++){
        read(u); read(v); read(c);
        add_flow(u,n+i,INF);
        add_flow(v,n+i,INF);
        add_flow(n+i,T,c);
        sum+=c;
    }
    printf("%d\n",sum-hlpp());
}

int main()
{
    while(read(n),read(m))solve();
    return 0;
}
```

4.13 最小费用最大流

4.13.1 SPFA

```
#include<bits/stdc++.h>
using namespace std;
//最小费用最大流spfa版，求最大费用只需要取相反数，结果取相反数即可。

struct node
{
    int to,pos,cap,val;
};
const int MAXM=10009;
const int MAXN=1009;
const int INF=0x3f3f3f3f;
int n,m,a[MAXN],s[MAXM],t[MAXM],c[MAXM];
int pre[MAXN],preedge[MAXN];
vector<node> E[MAXN];
int S,T;


void init(int n)
{
    for(int i=1;i<=n;i++)
    E[i].clear();
    S=0;T=n+2;
}
```

```
void addedge(int u,int v,int ca,int va)
{
   E[u].push_back((node){v,(int)E[v].size(),ca,va});
   E[v].push_back((node){u,(int)E[u].size()-1,0,-va});
}

int SPFA()
{
   queue<int> que;
   int vis[MAXN],dis[MAXN];
   memset(vis,0,sizeof(vis));
   memset(pre,-1,sizeof(pre));
   for (int i=S;i<=T;i++) dis[i]=INF;
   que.push(0);
   vis[0]=1;
   dis[0]=0;
   while (!que.empty())
   {
      int head=que.front();que.pop();
      vis[head]=0;
      for (int i=0;i<(int)E[head].size();i++)
      {
         node &tmp=E[head][i];
         if (tmp.cap>0 && dis[tmp.to]>dis[head]+tmp.val)
         {
            dis[tmp.to]=dis[head]+tmp.val;
            pre[tmp.to]=head;
            preedge[tmp.to]=i;
            if (!vis[tmp.to])
            {
               que.push(tmp.to);
               vis[tmp.to]=0;
            }
         }
      }
   }
      if (dis[T]==INF) return 0;else return 1;
   }

int minCostMaxflow()
{
   int flow=0;
   int ans=0;
   while (SPFA())
   {
      int f=INF;
      for (int i=T;pre[i]!=-1;i=pre[i])
      {
         node &tmp=E[pre[i]][preedge[i]];
         f=min(f,tmp.cap);
      }
      for (int i=T;pre[i]!=-1;i=pre[i])
      {
         node &tmp=E[pre[i]][preedge[i]];
         tmp.cap-=f;
         E[tmp.to][tmp.pos].cap+=f;
         ans+=f*tmp.val;
      }
      flow+=f;
   }
   return ans;
}

int p[MAXN];
int main()
{
   int n,m;
   while(~scanf("%d %d",&n,&m)){
      init(n);
      p[0]=0;p[n+1]=0;
      for(int i=1;i<=n;i++)
      scanf("%d",&p[i]);

      int u,v,c;
      for(int i=1;i<=m;i++){
         scanf("%d %d %d",&u,&v,&c);
         addedge(u,v+1,INF,c);
      }
      for(int i=1;i<=n+1;i++){
         int dt=p[i]-p[i-1];
         if(dt>=0) addedge(S,i,dt,0);
         else if(dt<0) addedge(i,T,-dt,0);
      }
      for(int i=1;i<=n;i++)
      addedge(i+1,i,INF,0);

      printf("%d\n",minCostMaxflow());
   }
   return 0;
}
```

4.13.2 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
typedef pair<int, int> pii;
const int maxn = 1e4;
const int inf = 0x3f3f3f3f;
struct edge {
   int to, cap, cost, rev;
   edge() {}
   edge(int to, int _cap, int _cost, int _rev) :to(to), cap(_cap), cost(_cost), rev(_rev) {}
};
int V, H[maxn + 5], dis[maxn + 5], PreV[maxn + 5], PreE[maxn + 5];
vector<edge> G[maxn + 5];
void init(int n) {
   V = n;
   for (int i = 0; i <= V; ++i)G[i].clear();
}
void AddEdge(int from, int to, int cap, int cost) {
   G[from].push_back(edge(to, cap, cost, G[to].size()));
   G[to].push_back(edge(from, 0, -cost, G[from].size() - 1));
}
int Min_cost_max_flow(int s, int t, int f, int& flow) {
   int res = 0; fill(H, H + 1 + V, 0);
   while (f) {
      priority_queue <pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>> > q;
      fill(dis, dis + 1 + V, inf);
      dis[s] = 0; q.push(pair<int, int>(0, s));
      while (!q.empty()) {
         pair<int, int> now = q.top(); q.pop();
         int v = now.second;
         if (dis[v] < now.first)continue;
         for (int i = 0; i < G[v].size(); ++i) {
            edge& e = G[v][i];
            if (e.cap > 0 && dis[e.to] > dis[v] + e.cost + H[v] - H[e.to]) {
               dis[e.to] = dis[v] + e.cost + H[v] - H[e.to];
               PreV[e.to] = v;
```

```
                PreE[e.to] = i;
                q.push(pair<int, int>(dis[e.to], e.to));
            }
        }
    }
    if (dis[t] == inf)break;
    for (int i = 0; i <= V; ++i)H[i] += dis[i];
    int d = f;
    for (int v = t; v != s; v = PrevV[v])d = min(d, G[PrevV[v]][PreE[v]].cap);
    f -= d; flow += d; res += d*H[t];
    for (int v = t; v != s; v = PrevV[v]) {
        edge& e = G[PrevV[v]][PreE[v]];
        e.cap -= d;
        G[v][e.rev].cap += d;
    }
  }
  return res;
}
int a[maxn];
int main()
{
  int t;
  scanf("%d",&t);
  while(t--)
  {
    int n,k;
    scanf("%d%d",&n,&k);
    for(register int i=1;i<=n;++i) scanf("%d",&a[i]);
    int ss=0,s=1,t=2*n+2,tt=2*n+3;
    init(tt+1);
    AddEdge(ss,s,k,0);
    AddEdge(t,tt,k,0);
    for(register int i=1;i<=n;++i)
    {
      AddEdge(s,i+1,1,0);
      AddEdge(i+1+n,t,1,0);
      AddEdge(i+1,i+1+n,1,-a[i]);
      for(register int j=i+1;j<=n;++j)
      {
        if(a[j]>=a[i])
        {
          AddEdge(1+i+n,1+j,1,0);
        }
      }
    }
    int ans=0;
    printf("%d\n",-Min_cost_max_flow(ss,tt,inf,ans));
  }
  return 0;
}
```

4.13.3 zkw 费用流

```
//zkw费用流
//适用于流量大，费用取值范围不大的图，或者是每次增广的路径段数少的图
//区间k覆盖问题
//数轴上有一些带权值的左闭右开区间，选出权和尽量大的一些区间，使得任意一个数最多被k个区间覆盖。
//对于权值为w的区间[u,v)加边u->v，容量为1，费用为-w。
//对所有相邻的点加边i->i+1，容量为k，费用为0。
//对最左点到最右点的最小费用最大流取负即为答案
#include <cstdio>
#include <queue>
#include <algorithm>
using namespace std;
typedef long long ll;
```

```
const int maxn=1005;
const int maxm=maxn*maxn;
const int INF=0x3f3f3f3f;

int n,k,V,S,T;

struct edge{
    int v, cap, cost, nxt;
    edge(){}
    edge(int _v,int _cap,int _cost,int _nxt):v(_v),cap(_cap),cost(_cost),nxt(_nxt){}
}e[maxm];

int head[maxn],cur[maxn],d[maxn], tot=1;
int cost,flow,dS;
bool vis[maxn];
inline void addedge(int u,int v,int cap,int cost){
    ++tot; e[tot]=edge(v,cap,cost,head[u]); head[u]=tot;
    ++tot; e[tot]=edge(u,0,-cost,head[v]); head[v]=tot;
}

inline int aug(int u,int f){
    if(u==T)return flow+=f,cost+=dS*f,f;
    vis[u]=true;
    int l=f;
    for(int i=cur[u];i;i=e[i].nxt){
        if(e[i].cap && !vis[e[i].v] && !e[i].cost){
            int tmp=aug(e[i].v,min(l,e[i].cap));
            e[i].cap-=tmp; e[i^1].cap+=tmp; l-=tmp;
            cur[u]=i;
            if(!l)return f;
        }
    }
    return f-l;
}

inline bool relabel(){
    for(int i=1;i<=V;i++)d[i]=INF; d[T]=0;
    deque<int>q; q.push_back(T);
    while(!q.empty()){
        int dt, u=q.front(); q.pop_front();
        for(int i=head[u];i;i=e[i].nxt){
            dt = d[u]-e[i].cost;
            if(e[i^1].cap && dt < d[e[i].v]){
                d[e[i].v]=dt;
                if(d[e[i].v]<d[q.empty()?0:q.front()])q.push_front(e[i].v);
                else q.push_back(e[i].v);
            }
        }
    }
    for(int u=1;u<=V;u++)
    for(int i=head[u];i;i=e[i].nxt)
    e[i].cost+=d[e[i].v]-d[u];
    dS+=d[S];
    return d[S]<INF;
}

void MinCostMaxFlow(){
    cost=0, flow=0; dS=0;
    while(relabel()){
        for(int i=1;i<=V;i++)cur[i]=head[i];
        do{
            for(int i=1;i<=V;i++)vis[i]=false;
        }while(aug(S,INF));
    }
}
```

```
void init(int N){
    V=N; for(int i=1;i<=V;i++)head[i]=0; tot=1;
}

int t[maxn<<1],m;
int a[maxn],b[maxn],w[maxn];
void solve(){
    scanf("%d %d",&n,&k); m=0;
    for(int i=1;i<=n;i++){
        scanf("%d %d %d",&a[i],&b[i],&w[i]);
        t[++m]=a[i]; t[++m]=b[i];
    }
    sort(t+1,t+1+m);
    m=unique(t+1,t+1+m)-(t+1);
    for(int i=1;i<=n;i++){
        a[i]=lower_bound(t+1,t+1+m,a[i])-t+1;
        b[i]=lower_bound(t+1,t+1+m,b[i])-t+1;
    }

    S=1,T=m+2; init(T);
    for(int i=1;i<T;i++) addedge(i,i+1,k,0);
    for(int i=1;i<=n;i++) addedge(a[i],b[i],1,-w[i]);
    MinCostMaxFlow();
    printf("%d\n",-cost);
}

int main()
{
    int _T; scanf("%d",&_T); for(int _=1;_<=_T;_++)solve();
    return 0;
}
```

4.14 SCC

```
int block,tp,id;
int low[maxn],dfn[maxn],ins[maxn];
int st[maxn],belong[maxn],sz[maxn],fat[maxn];
void tarjan(int u,int fa){
    low[u]=dfn[u]=++id;
    ins[u]=1;
    st[++tp]=u;
    fat[u]=fa;
    int k=0;
    for(int i=0;i<(int)G[u].size();i++){
        int to=G[u][i];
        if(to==fa && !k){
            k++;
            continue;
        }
        if(!dfn[to]){
            tarjan(to,u);
            low[u]=min(low[u],low[to]);
        }
        else if(ins[to])low[u]=min(low[u],dfn[to]);
    }
    if(low[u]==dfn[u]){
        block++;
        int to;
        do{
            to=st[tp--];
            ins[to]=0;
            belong[to]=block;
            sz[block]++;
        }while(u!=to);
    }
}

//使用前需初始化
void init(){
    for(int i=1;i<=n;i++)
        G[i].clear();
    tp=block=id=0;
    memset(dfn,0,sizeof(dfn));
    memset(ins,0,sizeof(ins));
}

void work()
{
    for(int i=1;i<=n;i++)
        if(!dfn[i])tarjan(i,i);
}
```

4.15 2-SAT

```
//(a_0,a_1) ,(b_0,b_1), 若a_i和b_i冲突, 则连接(a_i,b_{i^1}),(a_{i^1},b_i)
//表示选了a_i就必须选b_{i^1},然后跑tarjan缩点
//若有a_i和a_{i^1}同色, 则return false
//若需要输出方案, 则对于(a_i, b_{i+1}), 选择color小的
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=1e4+5,maxk=5005;

int n,k;
int id[maxn][5];
char s[maxn][5][5],ans[maxk];
bool vis[maxn];

struct Edge{ int v,nxt;}e[maxn*100];
int head[maxn],tot=1;
void addedge(int u,int v){
    e[tot].v=v; e[tot].nxt=head[u]; head[u]=tot++;
}

int dfn[maxn],low[maxn],color[maxn],stk[maxn],ins[maxn],top,dfs_clock,c;
void tarjan(int x){
    stk[++top]=x;
    ins[x]=1;
    dfn[x]=low[x]=++dfs_clock;
    for(int i=head[x];i;i=e[i].nxt){
        int v=e[i].v;
        if(!dfn[v]){
            tarjan(v);
            low[x]=min(low[x],low[v]);
        }
        else if(ins[v])low[x]=min(low[x],dfn[v]);
    }
    if(dfn[x]==low[x]){
        c++;
        do{
            color[stk[top]]=c;
            ins[stk[top]]=0;
        }while(stk[top--]!=x);
    }
}

int main()
{
    scanf("%d %d",&k,&n);
    for(int i=1;i<=n;i++){
        for(int j=1;j<=3;j++)scanf("%d%s",&id[i][j],s[i][j]);

        for(int j=1;j<=3;j++){
            for(int k=1;k<=3;k++){
                if(j==k)continue;
                int u=2*id[i][j]-(s[i][j][0]=='B');
                int v=2*id[i][k]-(s[i][k][0]=='R');
                addedge(u,v);
            }
        }
    }

    for(int i=1;i<=2*k;i++)
    if(!dfn[i])tarjan(i);

    for(int i=1;i<=2*k;i+=2)if(color[i]==color[i+1]){
        puts("-1");
        return 0;
    }

    for(int i=1;i<=2*k;i+=2){
        int f1=color[i],f2=color[i+1];
        if(vis[f1]){ans[(i+1)>>1]='R';continue;}
        if(vis[f2]){ans[(i+1)>>1]='B';continue;}
        if(f1<f2){
            vis[f1]=1; ans[(i+1)>>1]='R';
        }
        else{
            vis[f2]=1; ans[(i+1)>>1]='B';
        }
    }
    ans[k+1]=0;
    printf("%s\n",ans+1);
    return 0;
}
```

4.16 BCC-Point

```
// Created by calabash_boy on 18-10-10.
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+100;
int first[maxn],des[maxn*2],nxt[maxn*2],tot;
int bcc_cnt,cnt_n[maxn],cnt_e[maxn],bcc_no[maxn];
int dfn[maxn],low[maxn],dfs_clock;
int st[maxn*2],top;bool ok[maxn];
vector<int> ans;vector<int> temp;
int m,n;
inline void addEdge(int x,int y){
    tot++;des[tot] = y;
    nxt[tot] = first[x];first[x] = tot;
}
void input(){
    cin>>n>>m;
    for (int i=0;i<m;i++){
        int u,v;scanf("%d%d",&u,&v);
        addEdge(u,v);addEdge(v,u);
    }
}
void dfs(int u,int fa){
    dfn[u] = low[u] = ++dfs_clock;
    for (int t = first[u];t;t=nxt[t]){
        int v = des[t];
        if (v==fa)continue;
        if (!dfn[v]){
            st[top++] = t;dfs(v,u);
            low[u] = min(low[u],low[v]);
            if (low[v]>=dfn[u]){
                bcc_cnt++;ok[bcc_cnt] = true;
                temp.clear();
                while (true){
                    int tt = st[--top];
                    temp.push_back((tt+1)/2);
                    if (bcc_no[des[tt]]!=bcc_cnt){
                        bcc_no[des[tt]] = bcc_cnt;
                        cnt_n[bcc_cnt]++;
                    }else{
                        ok[bcc_cnt] = false;
                    }
                    cnt_e[bcc_cnt]++;
                    if (tt==t)break;
                }
                if (ok[bcc_cnt]&&temp.size()>1){
                    for (int i=0;i<temp.size();i++){
                        ans.push_back(temp[i]);
                    }
                }
            }
        }else if (dfn[v]<dfn[u]){
            st[top++] = t;
            low[u] = min(low[u],dfn[v]);
        }
    }
}
void solve(){
    for (int i=1;i<=n;i++){if (!dfn[i])dfs(i,-1);}
    sort(ans.begin(),ans.end());
    cout<<ans.size()<<endl;
    for (int i=0;i<ans.size();i++){printf("%d ",ans[i]);}
}
int main(){
    input();
    solve();
    return 0;
}
```

4.17 BCC-Edge

```
// Created by calabash_boy on 18-10-10.
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+100;
int first[maxn],nxt[maxn*2],from[maxn*2],des[maxn*2],isBrige[maxn*2],tot;
int dfn[maxn],low[maxn],dfs_clock;
int cnt_e[maxn],cnt_n[maxn];int bcc_cnt;
bool ok[maxn];vector <int> ans;int m,n;
inline void addEdge(int x,int y){
    tot++;
    des[tot] =y;from[tot] =x;
    nxt[tot] = first[x];first[x] = tot;
}
void input(){
    cin>>n>>m;
    for (int i=0;i<m;i++){
        int u,v;scanf("%d%d",&u,&v);
        addEdge(u,v);addEdge(v,u);
    }
}
void dfs(int u,int fa){
    dfn[u] = low[u] = ++dfs_clock;
    for (int t = first[u];t;t=nxt[t]){
        int v = des[t];if (v==fa)continue;
        if (!dfn[v]){
            dfs(v,u);
            low[u] = min(low[v],low[u]);
            if (dfn[u]<low[v]){
                isBrige[t] = true;
                if (t&1){isBrige[t+1] = true;}
                else{isBrige[t-1] = true;}
            }
        }else if (dfn[v]<dfn[u]){low[u] = min(low[u],dfn[v]);}
    }
}
void blood_fill(int x){
    dfn[x] = bcc_cnt;
    for (int t = first[x];t;t=nxt[t]){
        if (isBrige[t])continue;
        int v = des[t];
        if (!dfn[v]){blood_fill(v);}
    }
}
void check(){
    for (int i=1;i<=n;i++){cnt_n[dfn[i]]++;}
    for (int i=1;i<=tot;i++){
        if (isBrige[i]) continue;
        cnt_e[dfn[des[i]]]++;
    }
    for (int i=1;i<=bcc_cnt;i++){
        if (cnt_n[i]*2==cnt_e[i]){ok[i]=1;}
    }
}
void output(){
    for (int i=1;i<=tot;i+=2){
        if (isBrige[i])continue;
        if (ok[dfn[des[i]]])ans.push_back((i+1)/2);
    }
    sort(ans.begin(),ans.end());
    cout<<ans.size()<<endl;
    for (int i=0;i<ans.size();i++){printf("%d ",ans[i]);}
    }
    void solve(){
        for (int i=1;i<=n;i++){if (!dfn[i])dfs(i,-1);}
        memset(dfn,0,sizeof dfn);
        for (int i=1;i<=n;i++){
            if (!dfn[i]){
                bcc_cnt++;
                blood_fill(i);
            }
        }
        check();output();
}
int main(){
    input();
    solve();
    return 0;
}
```

4.18 树分治点分

```
// 满足dis(a,b)为素数的(a,b)的个数
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=2e5+5;
```

```cpp
const int INF=0x3f3f3f3f;
const double pi=acos(-1.0);

struct Complex{
    double r,i;
    Complex(double R=0,double I=0){r=R;i=I;};
    inline Complex operator +(const Complex& A){return Complex(r+A.r,i+A.i);}
    inline Complex operator -(const Complex& A){return Complex(r-A.r,i-A.i);}
    inline Complex operator *(const Complex& A){return Complex(r*A.r-i*A.i,r*A.i+i*A.r);}
};
int N;

void prework(Complex* a){
    int i,j,k;
    for(i=1,j=N/2;i<N-1;i++){
        if(i<j)swap(a[i],a[j]);
        k=N/2;
        while(j>=k){
            j-=k;
            k/=2;
        }
        if(j<k)j+=k;
    }
}

void FFT(Complex* a,int flag){
    Complex x,y;
    prework(a);
    for(int i=2;i<=N;i<<=1){
        Complex w,wk(cos(-2*pi*flag/i),sin(-2*pi*flag/i));
        for(int j=0;j<N;j+=i){
            w=Complex(1,0);
            for(int k=j;k<j+i/2;k++){
                x=a[k]; y=w*a[k+i/2];
                a[k]=x+y;
                a[k+i/2]=x-y;
                w=w*wk;
            }
        }
    }
    if(flag==-1)for(int i=0;i<N;i++)a[i].r/=N;
}

Complex a[maxn],b[maxn];
```

```cpp
struct edge{
    int v,w,nxt;
}e[maxn<<1];
int head[maxn], vis[maxn], tot;
void init(int n){
    for(int i=1;i<=n;i++)head[i]=vis[i]=0; tot=1;
}
inline void addedge(int u,int v,int w){
    e[tot]=(edge){v,w,head[u]}; head[u]=tot++;
    e[tot]=(edge){u,w,head[v]}; head[v]=tot++;
}

int n,k;
int sz[maxn],maxv[maxn],Max,rt;
int dis[maxn],dcnt;
ll ans;
```

```
void dfs_size(int u,int fa){
    sz[u]=1; maxv[u]=0;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(vis[v] || v==fa)continue;
        dfs_size(v,u);
        sz[u]+=sz[v]; maxv[u]=max(maxv[u],sz[v]);
    }
}

void dfs_root(int r,int u,int fa){
    maxv[u]=max(maxv[u],sz[r]-sz[u]);
    if(Max>maxv[u]){
        Max=maxv[u];
        rt=u;
    }
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(vis[v] || v==fa)continue;
        dfs_root(r,v,u);
    }
}

void dfs_dis(int u,int fa,int d){
    dis[dcnt++]=d;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v, w=e[i].w;
        if(vis[v] || v==fa)continue;
        dfs_dis(v,u,d+w);
    }
}

int prime[maxn],pcnt;
bool used[maxn];
void seive(){
    pcnt=0;
    for(int i=2;i<maxn;i++){
        if(!used[i])prime[pcnt++]=i;
        for(int j=0;j<pcnt;j++){
            ll nxt=1ll*prime[j]*i;
            if(nxt>=maxn)break;
            used[nxt]=1;
            if(i%prime[j]==0)break;
        }
    }
}

ll cnt[maxn],c[maxn];
int calc(int rt,int d){
    ll res=0; dcnt=0;
    dfs_dis(rt,-1,d);

    int ma=0;
    for(int i=0;i<dcnt;i++){
        ma=max(ma,dis[i]);
        c[dis[i]]++;
    }

    N=1; while(N<=2*ma)N<<=1;

    for(int i=0;i<N;i++){
        if(i<=ma)a[i]=Complex(c[i],0);
        else a[i]=Complex(0,0);
    }
    FFT(a,1);
    for(int i=0;i<N;i++)a[i]=a[i]*a[i];
    FFT(a,-1);

    for(int i=0;i<N;i++)cnt[i]=floor(a[i].r+0.5);
    for(int i=0;i<dcnt;i++)cnt[2*dis[i]]--;
    for(int i=0;i<N;i++)cnt[i]/=2;

    for(int i=0;i<pcnt;i++){
        if(prime[i]<=2*ma)res+=cnt[prime[i]];
        else break;
    }

    for(int i=0;i<dcnt;i++)c[dis[i]]--;

    return res;
}

void DFS(int u){
    Max=n; dfs_size(u,-1); dfs_root(u,u,-1);
    vis[rt]=1; ans+=calc(rt, 0);
    for(int i=head[rt];i;i=e[i].nxt){
        int v=e[i].v, w=e[i].w;
        if(vis[v])continue;
        ans-=calc(v,w);
        DFS(v);
    }
}

void solve(){
    init(n);
    int u,v;
    for(int i=1;i<=n-1;i++){
        scanf("%d %d",&u,&v);
        addedge(u,v,1);
    }
    ans=0;
    DFS(1);
    // cout<<ans<<endl;
    ll all=1ll*n*(n-1)/2;
    printf("%.10f\n",1.0*ans/all);
}

int main()
{
    seive();
    // freopen("in.txt","r",stdin);
    // int _T; scanf("%d",&_T); for(int _=1;_<=_T;_++)solve();
    while(~scanf("%d",&n))solve();
    // solve();
    return 0;
}
```

4.19 树分治边分

```
//一般仅对二叉树进行边分
//非二叉树则加点重构树使其为二叉树
#include<bits/stdc++.h>
using namespace std;
const int MX = 2e5 + 5;
const int MXE = 4e6 + 5;
struct Edge {
    int v, w, nxt, pre;
} E[MXE], edge[MXE];
int Head[MX], head[MX], rear, tot, tail[MX];
int mark[MX], sz[MX];
int N, n, cnt, rt, midedge, Max;
void init() {
   memset(head, -1, sizeof(head));
   tot = 0;
}
void INIT() {
   memset(Head, -1, sizeof(Head));
   rear = 0;
}
void add(int u, int v, int w) {
   edge[tot].v = v;
   edge[tot].w = w;
   edge[tot].nxt = head[u];
   head[u] = tot++;
}
void ADD(int u, int v, int w) {
   E[rear].v = v;
   E[rear].w = w;
   E[rear].nxt = Head[u];
   Head[u] = rear++;
}
void Delete(int u, int i) {
   if (Head[u] == i) Head[u] = E[i].nxt;
   else E[E[i].pre].nxt = E[i].nxt;
   if (tail[u] == i) tail[u] = E[i].pre;
   else E[E[i].nxt].pre = E[i].pre;
}
//保证每个点的度不超过3
void build(int u, int fa) {
   int father = 0;
   for (int i = head[u]; ~i; i = edge[i].nxt) {
      int v = edge[i].v, w = edge[i].w;
      if (v == fa) continue;
      if (father == 0) { //还没有增加子节点，直接连上
         ADD(u, v, w); ADD(v, u, w);
         father = u;
         build(v, u);
      } else { //已经有一个子节点，则创建一个新节点，把v连在新节点上
         mark[++N] = 0;
         ADD(N, father, 0); ADD(father, N, 0);
         father = N;
         ADD(v, father, w); ADD(father, v, w);
         build(v, u);
      }
   }
}
//nxt是下一条边的编号，pre是上一条边的编号
void get_pre() {
   memset(tail, -1, sizeof(tail));
   for (int i = 1; i <= N; i++) {
      for (int j = Head[i]; ~j; j = E[j].nxt) {
         E[j].pre = tail[i];
         tail[i] = j;
      }
   }
}
//重建一个图
void rebuild() {
   INIT();
   N = n;
   for (int i = 1; i <= N; i++) mark[i] = 1;
   build(1, 0);
   get_pre();
   init();
```

```
}

struct point {
   int u, dis;
   point() {}
   point(int _u, int _dis) {
      u = _u; dis = _dis;
   }
   bool operator<(const point& _A)const {
      return dis < _A.dis;
   }
};
struct node {
   int rt, midlen, ans; //根节点，中心边，答案(最长树链)
   int ls, rs; //左右子树编号
   priority_queue<point>q;
} T[2*MX];

//搜索每个子树大小
void dfs_size(int u, int fa, int dir) {
   add(u, rt, dir);
   //如果是白点，则压入根节点rt的队列，dist为到根的距离
   //队列中的点用来pt的父亲树的更新，pt节点不需要用到，
   //因此T[pt].rt是父亲树的中心边上的点
   if (mark[u]) T[rt].q.push(point(u, dir));
   sz[u] = 1;
   for (int i = Head[u]; ~i; i = E[i].nxt) {
      int v = E[i].v, w = E[i].w;
      if (v == fa) continue;
      dfs_size(v, u, dir + w);
      sz[u] += sz[v];
   }
}
//找中心边
void dfs_midedge(int u, int code) {
   if (max(sz[u], sz[T[rt].rt] - sz[u]) < Max) {
      Max = max(sz[u], sz[T[rt].rt] - sz[u]);
      midedge = code;
   }
   for (int i = Head[u]; ~i; i = E[i].nxt) {
      int v = E[i].v;
      if (i != (code ^ 1)) dfs_midedge(v, i);
   }
}
//更新
void PushUP(int rt) {
   T[rt].ans = -1;
   while (!T[rt].q.empty() && mark[T[rt].q.top().u] == 0) T[rt].q.pop();//弹出黑点
   int ls = T[rt].ls, rs = T[rt].rs; //ls为左儿子，rs为右儿子
   if (ls == 0 && rs == 0) { //没有左右儿子
      if (mark[T[rt].rt])T[rt].ans = 0;
   } else {
      if (T[ls].ans > T[rt].ans) T[rt].ans = T[ls].ans; //如果左儿子的结果大于右儿子
      if (T[rs].ans > T[rt].ans) T[rt].ans = T[rs].ans; //如果右儿子的结果大于左儿子
      if (!T[ls].q.empty() && !T[rs].q.empty()) //穿过中心边的
      T[rt].ans = max(T[rt].ans, T[ls].q.top().dis + T[rs].q.top().dis + T[rt].midlen);
   }
}
void DFS(int id, int u) {
   rt = id; Max = N; midedge = -1;
   T[id].rt = u;
   dfs_size(u, 0, 0);
   dfs_midedge(u, -1);
   if (~midedge) {
      //中心边的左右2点
```

```
        int p1 = E[midedge].v;
        int p2 = E[midedge ^ 1].v;
        //中心边长度
        T[id].midlen = E[midedge].w;
        //左右子树
        T[id].ls = ++cnt;
        T[id].rs = ++cnt;
        //删除中心边
        Delete(p1, midedge ^ 1);
        Delete(p2, midedge);
        DFS(T[id].ls, p1);
        DFS(T[id].rs, p2);
    }
    PushUP(id);
}
void update(int u) {
    mark[u] ^= 1;
    for (int i = head[u]; ~i; i = edge[i].nxt) {
        int v = edge[i].v, w = edge[i].w;
        if (mark[u] == 1) T[v].q.push(point(u, w));
        PushUP(v);
    }
}
int main() {
    scanf("%d", &n);
    init();
    for (int i = 1, u, v, w; i < n; i++) {
        scanf("%d%d%d", &u, &v, &w);
        add(u, v, w); add(v, u, w);
    }
    rebuild();
    DFS(cnt = 1, 1);
    char op[2]; int m, x;
    scanf("%d", &m);
    while (m--) {
        scanf("%s", op);
        if (op[0] == 'A') {
            if (T[1].ans == -1) printf("They have disappeared.\n");
            else printf("%d\n", T[1].ans);
        } else {
            scanf("%d", &x);
            update(x);
        }
    }
    return 0;
}
```

4.20 树同构 AHU algorithm

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=2e5+10;

int n;
struct Edge{
    int v,nxt;
}e[maxn<<1];
int head[maxn],sz[maxn],f[maxn],maxv[maxn],tag[maxn],tot,Max;
vector<int>center[2],L[maxn],subtree_tags[maxn];
void addedge(int u,int v){
    e[tot].v=v;e[tot].nxt=head[u];head[u]=tot++;
    e[tot].v=u;e[tot].nxt=head[v];head[v]=tot++;
}
```

```cpp
void dfs_size(int u,int fa){
    sz[u]=1; maxv[u]=0;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(v==fa)continue;
        dfs_size(v,u);
        sz[u]+=sz[v];
        maxv[u]=max(maxv[u],sz[v]);
    }
}

void dfs_center(int rt,int u,int fa,int id){
    maxv[u]=max(maxv[u],sz[rt]-sz[u]);
    if(Max>maxv[u]){
        center[id].clear();
        Max=maxv[u];
    }
    if(Max==maxv[u])center[id].push_back(u);
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(v==fa)continue;
        dfs_center(rt,v,u,id);
    }
}

int dfs_height(int u,int fa,int depth){
    L[depth].push_back(u); f[u]=fa;
    int h=0;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(v==fa)continue;
        h=max(h,dfs_height(v,u,depth+1));
    }
    return h+1;
}

void init(int n){
    for(int i=1;i<=2*n;i++)head[i]=0;
    tot=1; center[0].clear(); center[1].clear();

    int u,v;
    for(int i=1;i<=n-1;i++){
        scanf("%d %d",&u,&v);
        addedge(u,v);
    }
    dfs_size(1,-1);
    Max=n; dfs_center(1,1,-1,0);

    for(int i=1;i<=n-1;i++){
        scanf("%d %d",&u,&v);
        addedge(u+n,v+n);
    }
    dfs_size(1+n,-1);
    Max=n; dfs_center(1+n,1+n,-1,1);
}

bool cmp(int u,int v){
    return subtree_tags[u]<subtree_tags[v];
}

bool rootedTreeIsomorphism(int rt1,int rt2){
    for(int i=0;i<=2*n+1;i++)L[i].clear(),subtree_tags[i].clear();
    int h1=dfs_height(rt1,-1,0);
    int h2=dfs_height(rt2,-1,0);
    if(h1!=h2)return false;
    int h=h1-1;
    for(int j=0;j<(int)L[h].size();j++)tag[L[h][j]]=0;
    for(int i=h-1;i>=0;i--){
        for(int j=0;j<(int)L[i+1].size();j++){
            int v=L[i+1][j];
            subtree_tags[f[v]].push_back(tag[v]);
        }

        sort(L[i].begin(),L[i].end(),cmp);

        for(int j=0,cnt=0;j<(int)L[i].size();j++){
            if(j && subtree_tags[L[i][j]]!=subtree_tags[L[i][j-1]])++cnt;
            tag[L[i][j]]=cnt;
        }
    }
    return subtree_tags[rt1]==subtree_tags[rt2];
}

bool treeIsomorphism(){
    if(center[0].size()==center[1].size()){
        if(rootedTreeIsomorphism(center[0][0],center[1][0]))return true;
        if(center[0].size()>1)return rootedTreeIsomorphism(center[0][0],center[1][1]);
    }
    return false;
}

int main()
{
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%d",&n);
        init(n);
        puts(treeIsomorphism()?"YES":"NO");
    }
    return 0;
}
```

4.21 树哈希

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
const int maxn=4005;
const ull seed1=2333;
const ull seed2=1e6+7;
ull hval[maxn];

int n,Max,root;
int head[maxn],deg[maxn],sz[maxn],maxv[maxn],tot=1;
struct Edge{int v,nxt;}e[maxn<<1];
void addedge(int u,int v){
    e[tot].v=v; e[tot].nxt=head[u]; head[u]=tot++;
    e[tot].v=u; e[tot].nxt=head[v]; head[v]=tot++;
}
vector<int>center;
void dfs_size(int u,int fa,int rt){
    sz[u]=1; maxv[u]=0;
    for(int i=head[u];i;i=e[i].nxt){
        if(e[i].v==fa || e[i].v==rt)continue;
        dfs_size(e[i].v,u,rt);
        sz[u]+=sz[e[i].v];
        maxv[u]=max(maxv[u],sz[e[i].v]);
    }
}

void dfs_center(int rt,int u,int fa){
    maxv[u]=max(maxv[u],sz[rt]-sz[u]);
    if(Max>maxv[u]){
        center.clear(); center.push_back(u);
        root=u; Max=maxv[u];
    }
    else if(Max==maxv[u])center.push_back(u);
    for(int i=head[u];i;i=e[i].nxt){
        if(e[i].v!=fa)dfs_center(rt,e[i].v,u);
    }
}

void dfs_hash(int u,int fa,int rt){
    vector<ull>h;
    for(int i=head[u];i;i=e[i].nxt){
        if(e[i].v==fa || e[i].v==rt)continue;
        dfs_hash(e[i].v,u,rt);
        h.push_back(hval[e[i].v]);
    }
    sort(h.begin(),h.end());
    ull val=seed2;
    for(ull hv:h)val=val*seed1+hv;
    hval[u]=val*seed1+seed2;
}

int ans=-1;
void check(){
    vector<int>rts(center);
    for(int rt:rts){
        map<ull,int>mp;
        for(int i=head[rt];i;i=e[i].nxt){
            center.clear(); dfs_size(rt,-1,-1);
            Max=n; dfs_center(e[i].v,e[i].v,rt);
            set<ull>s;
            for(int rtt:center){
                dfs_size(rtt,-1,rt);
                dfs_hash(rtt,-1,rt);
                s.insert(hval[rtt]);
            }
            for(ull hv:s)mp[hv]++;
        }
        for(auto it:mp)if(it.second>=deg[rt]){
            ans=deg[rt];
            return;
        }
    }
}

int main()
{
    scanf("%d",&n);
    int u,v;
    for(int i=1;i<=n-1;i++){
        scanf("%d %d",&u,&v);
        addedge(u,v);
        deg[u]++; deg[v]++;
    }
    Max=n; dfs_size(1,-1,-1); dfs_center(1,1,-1);
    check();
    printf("%d\n",ans);
    return 0;
}
```

5 数学

5.1 FFT

```cpp
#include <bits/stdc++.h>

using namespace std;

const double pi=acos(-1.0);
const int maxn=8e5+233;

struct complex
{
    double r,i;
    complex(double R=0,double I=0){r=R;i=I;}
    inline complex operator +(const complex& A){return complex(r+A.r,i+A.i);}
    inline complex operator -(const complex& A){return complex(r-A.r,i-A.i);}
    inline complex operator *(const complex& A){return complex(r*A.r-i*A.i,r*A.i+i*A.r);}
}a[maxn];

int N,n,m;
int ma=0;

void init(){
    N=1;
    while(N<ma) N<<=1;
}

void prework(complex* a){
    int i,j,k;
    for(i=1,j=N/2;i<N-1;i++){
        if(i<j) swap(a[i],a[j]);
        k=N/2;
        while(j>=k){
            j-=k;
            k/=2;
        }
        if(j<k) j+=k;
    }
}

void FFT(complex* a,int flag){
    complex x,y;
    prework(a);
    for(int i=2;i<=N;i<<=1){
        complex w,wk(cos(-2*pi*flag/i),sin(-2*pi*flag/i));
        for(int j=0;j<N;j+=i){
            w=complex(1,0);
            for(int k=j;k<j+i/2;k++){
                x=a[k];
                y=w*a[k+i/2];
                a[k]=x+y;
                a[k+i/2]=x-y;
                w=w*wk;
            }
        }
    }
    if(flag==-1) for(int i=0;i<N;i++) a[i].r/=N;
}

int A[maxn];
long long cnt[maxn],suf[maxn];
double ans=0;

int main()
```

```
{
    int t;
    scanf("%d",&t);
    while(t--){
        memset(a,0,sizeof(a));
        memset(cnt,0,sizeof(cnt));
        scanf("%d",&n);
        ma=0;
        for(int i=0;i<n;i++){
            scanf("%d",&A[i]);
            a[A[i]].r++;
            ma=max(ma,A[i]);
        }
        sort(A,A+n);
        ma=ma*2+2;

        init();
        FFT(a,1);
        for(int i=0;i<N;i++) a[i]=a[i]*a[i];
        FFT(a,-1);

        for(int i=0;i<N;i++){
            cnt[i]=floor(a[i].r+0.5);
        }
        for(int i=0;i<n;i++){
            cnt[2*A[i]]--;
        }
        for(int i=0;i<N;i++){
            cnt[i]/=2;
        }
        suf[N]=0;
        for(int i=N-1;i>=0;i--){
            suf[i]=suf[i+1]+cnt[i];
        }

        long long tmp=0,sum=(long long)n*(n-1)*(n-2)/6;
        for(int i=0;i<n;i++){
            tmp+=suf[A[i]+1];

            tmp-=(long long)(n-1-i)*i;
            tmp-=n-1;
            tmp-=(long long)(n-1-i)*(n-i-2)/2;
        }

        ans=1.0*tmp/sum;
        printf("%.7f\n",ans);
    }
    return 0;
}
```

5.2 FWT

```
int rev=mod+1>>1;
void FWT(int a[],int n)
{
    for(int d=1;d<n;d<<=1)
    for(int m=d<<1,i=0;i<n;i+=m)
    for(int j=0;j<d;j++)
    {
        int x=a[i+j],y=a[i+j+d];
        a[i+j]=(x+y)%mod,a[i+j+d]=(x-y+mod)%mod;
        //xor:a[i+j]=x+y,a[i+j+d]=(x-y+mod)%mod;
        //and:a[i+j]=x+y;
        //or:a[i+j+d]=x+y;
    }
}

void UFWT(int a[],int n)
{
    for(int d=1;d<n;d<<=1)
    for(int m=d<<1,i=0;i<n;i+=m)
    for(int j=0;j<d;j++)
    {
        int x=a[i+j],y=a[i+j+d];
        a[i+j]=1LL*(x+y)*rev%mod,a[i+j+d]=(1LL*(x-y)*rev%mod+mod)%mod;
        //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
        //and:a[i+j]=x-y;
        //or:a[i+j+d]=y-x;
    }
}
void solve(int a[],int b[],int n)
{
    FWT(a,n);
    FWT(b,n);
    for(int i=0;i<n;i++)
    a[i]=1LL*a[i]*b[i]%mod;
    UFWT(a,n);
}
```

5.3 NTT

```
//常用素数：1004535809,998244353,原根都为3
#include<cstdio>
#define getchar() (p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 1<<21, stdin), p1 == p2) ? EOF : *p1++)
#define swap(x,y) x ^= y, y ^= x, x ^= y
#define LL long long
const int MAXN = 3 * 1e6 + 10, P = 998244353, G = 3, Gi = 332748118;
char buf[1<<21], *p1 = buf, *p2 = buf;
inline int read() {
    char c = getchar(); int x = 0, f = 1;
    while(c < '0' || c > '9') {if(c == '-') f = -1; c = getchar();}
    while(c >= '0' && c <= '9') x = x * 10 + c - '0', c = getchar();
    return x * f;
}
int N, M, limit = 1, L, r[MAXN];
LL a[MAXN], b[MAXN];
inline LL fastpow(LL a, LL k) {
    LL base = 1;
    while(k) {
        if(k & 1) base = (base * a ) % P;
        a = (a * a) % P;
        k >>= 1;
    }
    return base % P;
}
inline void NTT(LL *A, int type) {
    for(int i = 0; i < limit; i++)
    if(i < r[i]) swap(A[i], A[r[i]]);
    for(int mid = 1; mid < limit; mid <<= 1) {
        LL Wn = fastpow( type == 1 ? G : Gi , (P - 1) / (mid << 1));
        for(int j = 0; j < limit; j += (mid << 1)) {
            LL w = 1;
            for(int k = 0; k < mid; k++, w = (w * Wn) % P) {
            int x = A[j + k], y = w * A[j + k + mid] % P;
            A[j + k] = (x + y) % P,
            A[j + k + mid] = (x - y + P) % P;
        }
    }
}
}
```

```
int main() {
    N = read(); M = read();
    for(int i = 0; i <= N; i++) a[i] = (read() + P) % P;
    for(int i = 0; i <= M; i++) b[i] = (read() + P) % P;
    while(limit <= N + M) limit <<= 1, L++;
    for(int i = 0; i < limit; i++) r[i] = (r[i >> 1] >> 1) | ((i & 1) << (L - 1));
    NTT(a, 1);NTT(b, 1);
    for(int i = 0; i < limit; i++) a[i] = (a[i] * b[i]) % P;
    NTT(a, -1);
    LL inv = fastpow(limit, P - 2);
    for(int i = 0; i <= N + M; i++)
    printf("%d ", (a[i] * inv) % P);
    return 0;
}
```

5.4 博弈论

5.4.1 定理

```
/*
公平组合博弈（ICG）
1.巴什博奕(Bash Game)
只有一堆n个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取m个。最后取光者得胜。
显然，如果n=m+1，1那么由于一次最多只能取m个，所以，无论先取者拿走多少个，后取者都能够一次拿走剩余的物品，后者取胜。因此我们发现了如何取胜的法则：每个回合时m+1个，如果n=（m+1)*r+s，（r为任意自然数，s≤m),那么先取者要拿走s个物品，如果后取者拿走k（≤m)个，那么先取者再拿走m+1-k个，结果剩下（m+1）（r-1）个，以后保持这样的取法，那么先取者肯定获胜。总之，要保持给对手留下（m+1）的倍数，就能最后获胜。
这个游戏还可以有一种变相的玩法：两个人轮流报数，每次至少报一个，最多报十个，谁能报到100者胜。

2.斐波那契博弈（Fibonaci's Game）
有一堆个数为n的石子，游戏双方轮流取石子，满足：
1)先手不能在第一次把所有的石子取完；
2)之后每次可以取的石子数介于1到对手刚取的石子数的2倍之间（包含1和对手刚取的石子数的2倍）。
引理：任何正整数可以表示为若干个不连续的Fibonacci数之和
结论：先手胜当且仅当n不是斐波那契数，否则必败
策略：比如，我们要分解83，注意到83被夹在55和89之间，于是把83可以写成83=55+28；然后再想办法分解28，28被夹在21和34之间，于是28=21+7；依此类推 7=5+2。

如果n=83，我们看看这个分解有什么指导意义：假如先手取2颗，那么后手无法取5颗或更多，而5是一个Fibonacci数，如果猜测正确的话，（面临这5颗的先手实际上是整个游戏的后手）那么一定是先手取走这5颗石子中的最后一颗，而这个我们可以通过第二类归纳法来绕过，同样的道理，接下去先手取走接下来的后21颗中的最后一颗，再取走后55颗中的最后一颗，那么先手赢。

反过来如果n是Fibonacci数，比如n=89：记先手一开始所取的石子数为y，若y>=34颗（也就是89的向前两项），那么一定后手赢，因为89-34=55=34+21<2*34，所以只需要考虑先手第一次取得石子数y<34的情况即可，所以现在剩下的石子数x介于55到89之间，它一定不是一个Fibonacci数，于是我们把x分解成Fibonacci数：x=55+f[i]+…+f[j]，若，如果f[j]<=2y，那么对B就是面临x局面的先手，所以根据之前的分析，B只要先取f[j]个即可，以后再按之前的分析就可保证必胜。

3.威佐夫博奕（Wythoff Game）
有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。
这种情况下是颇为复杂的。我们用（ak，bk）（ak ≤ bk ,k=0，1，2，…,n)表示两堆物品的数量并称其为局势，如果甲面对（0，0），那么甲已经输了，这种局势我们称为奇异局势。前几个奇异局势是：（0，0）、（1，2）、（3，5）、（4，7）、（6，10）、（8，13）、（9，15）、（11，18）、（12，20）。
可以看出,a0=b0=0,ak是未在前面出现过的最小自然数,而 bk= ak + k，奇异局势有
如下三条性质：
1.任何自然数都包含在一个且仅有一个奇异局势中。
由于ak是未在前面出现过的最小自然数，所以有ak > ak-1 ，而 bk= ak + k > ak-1 + k-1 = bk-1 > ak-1 。所以性质1。成立。
2.任意操作都可将奇异局势变为非奇异局势。
事实上，若只改变奇异局势（ak，bk）的某一个分量，那么另一个分量不可能在其他奇异局势中，所以必然是非奇异局势。如果使（ak，bk）的两个分量同时减少，则由于其差不变，且不可能是其他奇异局势的差，因此也是非奇异局势。
3.采用适当的方法，可以将非奇异局势变为奇异局势。
从如上性质可知，两个人如果都采用正确操作，那么面对非奇异局势，先拿者必胜；反之，则后拿者取胜。
4.（Betty 定理）：如果存在正无理数 A，B 满足 1/A + 1/B = 1，那么集合 P = { [At]，t ▯ Z+}、Q = { [Bt]，t ▯ Z+} 恰为集合 Z+ 的一个划分，即：P ▯ Q = Z+，P ∩ Q = ø。
5.上述矩阵中每一行第一列的数为 [Φi]，第二列的数为 [(Φ + 1)i]，其中 Φ = (sqrt(5) + 1) / 2 为黄金分割比。
```

```
那么任给一个局势（a，b），怎样判断它是不是奇异局势呢？我们有如下公式:
ak =[k（1+√5）/2]，bk= ak + k （k=0，1，2，…,n 方括号表示取整函数)
奇妙的是其中出现了黄金分割数（1+√5）/2 = 1.618…,因此,由ak，bk组成的矩形近似为黄金矩形，由于2/（1+√5）=（√5-1）/2，可以
    先求出j=[a（√5-1）/2]，若a=[（1+√5）/2]，那么a = aj，bj = aj + j，若不等于，那么a = aj+1，bj+1 = aj+1+ j + 1，若
    都不是，那么就不是奇异局势。然后再按照上述法则进行，一定会遇到奇异局势。（poj 1067）

4.尼姆博奕（Nimm Game）
有n堆各若干个物品，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限，最后取光者得胜。
结论:将n堆物品数量全部异或 : 为零则必败,否则必胜(先手)

5.阶梯Nim
结论：奇数阶异或和为0则必胜，反之必败。偶数阶不考虑。

6.SJ定理：对于任意一个 Anti-SG 游戏，如果我们规定当局面中所有的单一游戏的 SG 值为 0 时，游戏结束，则先手必胜当且仅当:
（1）游戏的 SG 函数不为0且游戏中某个单一游戏的 SG 函数大于 1;
（2）游戏的 SG 函数为0且游戏中没有单一游戏的 SG 函数大于 1
*/
```

5.4.2 SG 函数

```
/*
Sprague-Grundy Theorem:
g(G)=g(G1)^g(G2)^...^g(Gn)。也就是说，游戏的和的SG函数值是它的所有子游戏的SG函数值的异或。
*/

//将一个堆拆分成两个不相等的堆
void SG(int x){
   memset(sg,-1,sizeof(sg));
   memset(vis,0,sizeof(vis));
   int tmp=0;
   sg[1]=0;
   for(int i=1;i<=x;++i,++tmp){
      for(int j=1;j<=i/2;++j){
         if(j==i-j)continue;
         vis[sg[i-j]^sg[j]]=tmp;
      }
      for(int j=0;j<=x;++j){
         if(vis[j]!=tmp){
            sg[i]=j;
            break;
         }
      }
   }
}
//Nim + 最多拿一半
void SG(int x){
   memset(sg,-1,sizeof(sg));
   memset(vis,0,sizeof(vis));
   int tmp=0;
   for(int i=1;i<=x;++i,++tmp){
      for(int j=1;j<=i/2;++j){
         vis[sg[i-j]]=tmp;
      }
      for(int j=0;j<=x;++j){
         if(vis[j]!=tmp){
            sg[i]=j;
            break;
         }
      }
   }
}


//Nim + 拿走最后一的的赢 （anti-nim)
#include<bits/stdc++.h>
```

```
using namespace std;
typedef long long ll;
const int maxn=1e5+233;
int n,m;
int a[maxn];
int main()
{
    int t,tim=1;
    scanf("%d",&t);
    while(t--){
        int u=0,v,cnt=0;
        scanf("%d",&n);
        for(int i=0;i<n;i++){
            scanf("%d",&v);
            u^=v;
            cnt+=(v==1);
        }
        if((cnt==n && u==0) || (cnt<n && u))
        printf("Case %d: Alice\n",tim++);
        else
        printf("Case %d: Bob\n",tim++);
    }
    return 0;
}
```

5.5 莫比乌斯反演

$$TBA \quad (3)$$

5.5.1 线性筛

```
/* x in [1,N]; y in [1,M] (x,y) = 1 */
#include<cstdio>
#include<vector>
using namespace std;
const int maxn = 1e5+100;
typedef long long ll;
bool used[maxn];
vector<int> prime;
ll mu[maxn];
void sieve(){
    mu[1] = 1;
    for (int i=2;i<maxn;i++){
        if(!used[i]){
            prime.push_back(i);
            mu[i] = -1;
        }
        for (int j = 0;j<prime.size();j++){
            long long nxt = 1ll* prime[j] * i;
            if(nxt >= maxn)break;
            used[nxt] = 1;
            if (i % prime[j] == 0){
            mu[nxt] = 0;
            break;
        }else{
            mu[nxt] = -mu[i];
        }
    }
}
}
ll work(int n,int m){
    ll ans = 0;
    int top = min(n,m);
    for (int i=1;i<=top;i++){
        ans += 1ll * mu[i] * (n/i) * (m/i);
    }
    return ans;
}
int main(){
    sieve();
    int T;
    scanf("%d",&T);
    for (int Case = 1;Case <= T;Case ++){
        int a,b,n,m,k;
        scanf("%d%d%d%d%d",&a,&n,&b,&m,&k);
        if(k == 0){
            printf("Case %d: 0\n",Case);
            continue;
        }
        n/=k;
        m/=k;
        printf("Case %d: %lld\n",Case,work(n,m) - work(min(n,m),min(n,m))/2);
    }
    return 0;
}
```

5.6 拉格朗日插值法

已知 P(i),0<=i<=n 的值 (特殊情况)

$$P(x)=\sum_{i=0}^{n}(-1)^{n-i}P(i)\frac{x(x-1)(x-2)...(x-n)}{(n-i)!i!(x-i)}$$

已知任意 n+1 个给定的点 x_i 以及值 $P(x_i)$ (一般情况)

$$P(x)=\sum_{i=0}^{n}P(x_i)\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$$

5.7 斐波那契公式

$$fibonacci_n=-\frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n+\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n$$

$$fibonacci_n=\begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}^{n-1}$$

5.8 生成函数

$$\frac{1}{1-x^a}=1+x^a+x^{2a}+x^{3a}...$$

$$\frac{1}{(1-x)^2}=1+2x+3x^2+4x^3...$$

$$\frac{1}{(1-x)^k}=\sum_{i}^{\infty}C_{i+k-1}^{k-1}x^i$$

5.9 Stern-Brocot Tree

满足 BST 性质
第 n 层为 n 阶 Farey 级数

$$L=\frac{a}{b},R=\frac{c}{d}$$
$$\text{新的中间项}M=\frac{a+c}{b+d} \tag{4}$$

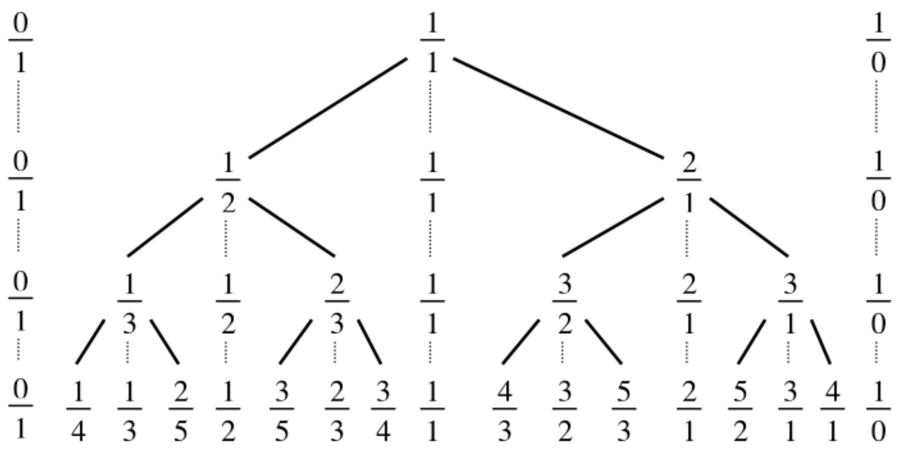
Figure 1: Part of the Stern-Brocot Tree.

6 其他

6.1 动态规划

6.1.1 区间 DP

```
for(int l=2;l<=n;l++)
{
    for(int i=1;i<=n-l+1;i++)
    {
        for(int k=i;k<=i+l-2;k++)
        {
            f[i][i+l-1]=min(f[i][i+l-1],f[i][k]+f[k+1][i+l-1]+sum[i+l-1]-sum[i-1]);
            //左区间代价+右区间代价+决策值
        }
    }
}
ans=f[1][n]; //区间[1,n]的最优解
```

6.1.2 数位 DP

```
//不含4和62连号
long long dfs(int pos,int six,int jud)
{
    if(pos==0) return 1;
    int sz=jud?digit[pos]:9;
    if(!jud&&dp[pos][six]!=-1) return dp[pos][six];
    long long ans=0;
    for(int i=0;i<=sz;i++)
    {
        if(i==4) continue;
        if(six&&(i==2)) continue;
        ans+=dfs(pos-1,i==6,jud&&(i==sz));
    }
    if(!jud) dp[pos][six]=ans;
    return ans;
}
long long f(long long x)
{
    int pos=0;
    while(x)
    {
        digit[++pos]=x%10;
        x/=10;
    }
    return dfs(pos,0,1);
}
```

6.1.3 单调队列优化

$$f_{i,j} = max\left\{f_{i-1,k} + b_i - |a_i - j|\right\}$$
$$f_{i,j} = max\left\{f_{i-1,k}\right\} + b_i - |a_i - j| \tag{5}$$

```
#include<stdio.h>
#include<string>
using namespace std;
long long f[150010][2];
int p[305],b[305],t[305];
int Abs(int x){
    return x>=0?x:-x;
}
struct Node{
    int id;
    long long val;
}nd[150010];
int main(){
    int n,m;
    long long d;
    scanf("%d%d%I64d",&n,&m,&d);
    for(int i=1;i<=m;i++)scanf("%d%d%d",&p[i],&b[i],&t[i]);
    long long mx;
    for(int i=1;i<=m;i++){
        int hd=1,tl=1;nd[tl].id=1,nd[tl++].val=f[1][(i-1)&1];
        for(int j=1;j<=n&&j<=d*(t[i]-t[(i-1)]);j++){
            while(hd<tl&&nd[tl-1].val<=f[j][(i-1)&1])tl--;
            nd[tl].val=f[j][(i-1)&1],nd[tl++].id=j;
        }
        for(int j=1;j<=n;j++){
            if(j+d*(t[i]-t[i-1])<=n){
                while(hd<tl&&nd[tl-1].val<=f[j+d*(t[i]-t[i-1])][(i-1)&1])tl--;
                nd[tl].val=f[j+d*(t[i]-t[i-1])][(i-1)&1],nd[tl++].id=j+d*(t[i]-t[i-1]);
            }
            if(nd[hd].id+d*(t[i]-t[i-1])<j)hd++;
            f[j][i&1]=nd[hd].val+b[i]-Abs(p[i]-j);
            if(i==m)if(j==1)mx=f[j][i&1];else mx=max(f[j][i&1],mx);
        }
    }
    printf("%I64d\n",mx);
}
```

6.1.4 斜率优化

$$f_i = min\left\{f_j + (sum_i - sum_j + i - j - 1 - L)^2\right\}$$
$$s_i = sum_i + i, L' = L + 1$$
$$f_i = f_j + (s_i - s_j - L')^2 = f_j + (s_i - L')^2 - 2(s_i - L') * s_j + s_j^2$$
$$f_j + s_j^2 = 2(s_i - L') \times s_j + f_i - (s_i - L')^2$$
$$y = k \times x + b \tag{6}$$

```
#include<stdio.h>
#include<string>
using namespace std;
long long c[50010],s[50010],b,x[50010],y[50010],f[50010];
int nd[50010];
long long ans(int i,int j){
    return f[j]+(s[i]-s[j]-b)*(s[i]-s[j]-b);
}
bool cmp(int i,int j,int k){
    return (y[k]-y[j])*(x[j]-x[i])<=(y[j]-y[i])*(x[k]-x[j]);
}
int main(){
    int n,l;
    scanf("%d%d",&n,&l);
    for(int i=1;i<=n;i++)scanf("%lld",&c[i]);
    for(int i=1;i<=n;i++)s[i]=s[i-1]+c[i]+1;
    b=l+1;
    x[0]=y[0]=0;
    int hd=1,tl=1;nd[tl++]=0;
    for(int i=1;i<=n;i++){
        while(hd+1<tl&&ans(i,nd[hd])>=ans(i,nd[hd+1]))hd++;
        f[i]=ans(i,nd[hd]);
        y[i]=f[i]+s[i]*s[i];
        x[i]=s[i];
        while(hd+1<tl&&cmp(nd[tl-2],nd[tl-1],i))tl--;
        nd[tl++]=i;
    }
    printf("%lld",f[n]);
}
```

6.2 整体二分

6.2.1 基本思路

```
void solve(int l, int r, int L, int R)
{
    if(l>r || L>R) return;
    if(l==r)
    {
        for(int i=L;i<=R;i++) if(q[i].ty) ans[q[i].pos]=l; //如果q[i].ty==1是询问，就把答案丢进去
        return;
    }
    int mid=(l+r)>>1, cnt1=0, cnt2=0; //二分答案
    for(int i=L;i<=R;i++)
    if(q[i].ty)
    {
        int tmp=query(q[i].y)-query(q[i].x-1); //求bit上询问区间的sum
        if(tmp>=q[i].k) q1[++cnt1]=q[i]; //<=k就丢左边
        else q[i].k-=tmp, q2[++cnt2]=q[i]; //>k就更新k后丢右边
    }
    else
    {
        if(q[i].x<=mid) add(q[i].pos, q[i].y), q1[++cnt1]=q[i]; //如果被修改数<=mid，就更新bit并丢到左边
        else q2[++cnt2]=q[i];
    }
    for(int i=1;i<=cnt1;i++) if(!q1[i].ty) add(q1[i].pos, -q1[i].y); //如果是修改，删去在bit上的值
    for(int i=1;i<=cnt1;i++) q[L+i-1]=q1[i]; //丢到左边~
    for(int i=1;i<=cnt2;i++) q[L+cnt1+i-1]=q2[i]; //丢到右边~
    solve(l, mid, L, L+cnt1-1); solve(mid+1, r, L+cnt1, R); //分治~
}
```

6.2.2 区间带修改第 k 小

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=5e5+5;
const int INF=1e9;
int n,m;
int a[maxn],ans[maxn];

struct BIT{
    int c[maxn];
    inline int lb(int x){return x&(-x);}
    inline void add(int x,int d){for(;x<=n;x+=lb(x))c[x]+=d;}
    inline int getsum(int x){int r=0;for(;x;x-=lb(x))r+=c[x];return r;}
    inline int getsum(int l,int r){return getsum(r)-getsum(l-1);}
}bit;

struct node{
    int x,y,k;
    int id,type;
}nd[maxn],tmp1[maxn],tmp2[maxn];

void solve(int l,int r,int L,int R)
{
    if(l>r || L>R)return;
    if(l==r){
        for(int i=L;i<=R;i++)if(nd[i].type)ans[nd[i].id]=l;
        return;
    }
    int mid=(l+r)>>1,t1=0,t2=0;
    for(int i=L;i<=R;i++){
        if(nd[i].type){
            int tmp=bit.getsum(nd[i].x,nd[i].y);
            if(tmp>=nd[i].k)tmp1[++t1]=nd[i];
            else nd[i].k-=tmp,tmp2[++t2]=nd[i];
        }
        else{
            if(nd[i].x<=mid)bit.add(nd[i].id,nd[i].y),tmp1[++t1]=nd[i];
            else tmp2[++t2]=nd[i];
        }
    }
    for(int i=1;i<=t1;i++)if(!tmp1[i].type)bit.add(tmp1[i].id,-tmp1[i].y);
    for(int i=1;i<=t1;i++)nd[L+i-1]=tmp1[i];
    for(int i=1;i<=t2;i++)nd[L+t1+i-1]=tmp2[i];
    solve(l,mid,L,L+t1-1); solve(mid+1,r,L+t1,R);
}

void mainwork()
{
    int tot=0;
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        nd[++tot]=(node){a[i],1,0,i,0};
    }

    char op[5];
    int x,y,z,id=0;
    for(int i=1;i<=m;i++){
        scanf("%s",op);
        if(op[0]=='Q'){
            scanf("%d %d %d",&x,&y,&z);
            nd[++tot]=(node){x,y,z,++id,1};
        }
        else{
            scanf("%d %d",&x,&y);
            nd[++tot]=(node){a[x],-1,0,x,0};
            a[x]=y;
            nd[++tot]=(node){a[x],1,0,x,0};
        }
    }
    solve(-INF,INF,1,tot);
    for(int i=1;i<=id;i++)printf("%d\n",ans[i]);
}

int main()
{
    int T;
    scanf("%d",&T);
    while(T--)mainwork();
    return 0;
}
```

6.2.3 树上带修改第 k 小

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=8e4+5;
int n,q;
int w[maxn],ans[maxn];

struct Operate{
    int id,k,a,b,val,type;
}op[maxn<<1],tmp1[maxn<<1],tmp2[maxn<<1];

struct BIT{
    int c[maxn];
    inline int lb(int x){return x&(-x);}
    inline void add(int x,int d){for(;x<=n;x+=lb(x))c[x]+=d;}
    inline int getsum(int x){int r=0;for(;x;x-=lb(x))r+=c[x];return r;}
    inline int getsum(int l,int r){return getsum(r)-getsum(l-1);}
}bit;

//graph
struct Edge{
    int v,nxt;
}e[maxn<<1];
int head[maxn],tot;
void init(){
    tot=1;
    memset(head,0,sizeof(head));
}
void addedge(int u,int v){
    e[tot].v=v;e[tot].nxt=head[u];head[u]=tot++;
    e[tot].v=u;e[tot].nxt=head[v];head[v]=tot++;
}

//heavy-light
int sz[maxn],son[maxn],fa[maxn],h[maxn],pos[maxn],top[maxn],cnt;
void dfs1(int u,int f){
    sz[u]=1;son[u]=0;fa[u]=f;h[u]=h[f]+1;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(v==f)continue;
        dfs1(v,u);
        sz[u]+=sz[v];
        if(sz[son[u]]<sz[v])son[u]=v;
    }
}
void dfs2(int u,int f,int k){
    top[u]=k;
    pos[u]=++cnt;
    if(son[u])dfs2(son[u],u,k);
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].v;
        if(v==f || v==son[u])continue;
        dfs2(v,u,v);
    }
}
int LCA(int u,int v){
    while(top[u]!=top[v]){
        if(h[top[u]]<h[top[v]])swap(u,v);
        u=fa[top[u]];
    }
    if(h[u]>h[v])swap(u,v);
    return u;
}
int query(int u,int v){
    int res=0;
    while(top[u]!=top[v]){
        if(h[top[u]]<h[top[v]])swap(u,v);
        res+=bit.getsum(pos[top[u]],pos[u]);
        u=fa[top[u]];
    }
    if(h[u]>h[v])swap(u,v);
    res+=bit.getsum(pos[u],pos[v]);
    return res;
}

void solve(int l,int r,int L,int R)
{
    if(l>r || L>R)return;
    if(l==r){
        for(int i=L;i<=R;i++)
        if(op[i].type==0 && ans[op[i].id]!=-1)ans[op[i].id]=l;
        return;
    }
    int mid=(l+r)>>1,t1=0,t2=0;
    for(int i=L;i<=R;i++){
        if(op[i].type==0){
            int tmp=query(op[i].a,op[i].b);
            if(tmp>=op[i].k)tmp2[++t2]=op[i];
            else op[i].k-=tmp,tmp1[++t1]=op[i];
        }
        else{
            if(op[i].b>mid)bit.add(pos[op[i].a],op[i].val),tmp2[++t2]=op[i];
            else tmp1[++t1]=op[i];
        }
    }
    for(int i=1;i<=t2;i++)if(tmp2[i].type==1 && tmp2[i].b>mid)
    bit.add(pos[tmp2[i].a],-tmp2[i].val);
    for(int i=1;i<=t1;i++)op[L+i-1]=tmp1[i];
    for(int i=1;i<=t2;i++)op[L+t1+i-1]=tmp2[i];
    solve(l,mid,L,L+t1-1); solve(mid+1,r,L+t1,R);
}

int main()
{
    int nn=0;
    scanf("%d %d",&n,&q);
    for(int i=1;i<=n;i++){
        scanf("%d",&w[i]);
        op[++nn]=(Operate){0,0,i,w[i],1,1};
    }

    int u,v;
    init();
    for(int i=1;i<=n-1;i++){
        scanf("%d %d",&u,&v);
        addedge(u,v);
    }
    dfs1(1,0);dfs2(1,0,1);

    int k,a,b,id=0;
    for(int i=1;i<=q;i++){
        scanf("%d %d %d",&k,&a,&b);
        if(k){
            op[++nn]=(Operate){++id,k,a,b,0,0};
            int lca=LCA(a,b);
            int len=h[a]+h[b]-h[lca]-h[fa[lca]];
            if(len<k)ans[id]=-1;
        }
        else{
            op[++nn]=(Operate){0,k,a,w[a],-1,1};
            w[a]=b;
            op[++nn]=(Operate){0,k,a,w[a],1,1};
        }
    }

    solve(1,1e8,1,nn);

    for(int i=1;i<=id;i++){
        if(ans[i]==-1)printf("invalid request!\n");
        else printf("%d\n",ans[i]);
    }
    return 0;
}
```

6.3 cdq 分治

6.3.1 3 维偏序

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=1e5+5;
const int maxv=2e5+5;

int n,nn,k;
int cnt[maxn],ans[maxv];

struct BIT{
    int c[maxv];
    inline int lowbit(int x){return x&(-x);}
    inline void add(int x,int d){for(;x<=k;x+=lowbit(x))c[x]+=d;}
    inline int getsum(int x){int r=0;for(;x;x-=lowbit(x))r+=c[x];return r;}
}bit;

struct node{
    int id,a,b,c,cnt;
    bool operator == (const node& x){
        return a==x.a && b==x.b && c==x.c;
    }
    inline bool operator < (const node &x)const{
        if(a!=x.a)return a<x.a;
        if(b!=x.b)return b<x.b;
        return c<x.c;
    }
}nd[maxn],tmp[maxn];

void cdq(int l,int r){
    if(l>=r)return;
    int mid=(l+r)/2;
    cdq(l,mid);cdq(mid+1,r);

    //merge sort
    int i,j,k;
    for(i=l,j=mid+1,k=l;i<=mid && j<=r;){
        if(nd[i].b<=nd[j].b){
            bit.add(nd[i].c,nd[i].cnt);
            tmp[k++]=nd[i++];
        }
        else{
            cnt[nd[j].id]+=bit.getsum(nd[j].c);
            tmp[k++]=nd[j++];
        }
    }
    while(i<=mid){
        bit.add(nd[i].c,nd[i].cnt);
        tmp[k++]=nd[i++];
    }
    while(j<=r){
        cnt[nd[j].id]+=bit.getsum(nd[j].c);
        tmp[k++]=nd[j++];
    }

    for(i=l;i<=mid;i++)bit.add(nd[i].c,-nd[i].cnt);

    for(i=l;i<=r;i++)nd[i]=tmp[i];
}

int main()
{
    scanf("%d %d",&n,&k);
    for(int i=1;i<=n;i++)scanf("%d %d %d",&nd[i].a,&nd[i].b,&nd[i].c);
    sort(nd+1,nd+1+n);

    tmp[++nn]=nd[1];
    tmp[nn].id=nn; tmp[nn].cnt=1;
    for(int i=2;i<=n;i++){
        if(tmp[nn]==nd[i])
        ++tmp[nn].cnt;
        else{
            tmp[++nn]=nd[i];
            tmp[nn].id=nn; tmp[nn].cnt=1;
        }
    }
    for(int i=1;i<=nn;i++)nd[i]=tmp[i];

    cdq(1,nn);

    for(int i=1;i<=nn;i++){
        cnt[nd[i].id]+=nd[i].cnt-1;
        ans[cnt[nd[i].id]]+=nd[i].cnt;
    }

    for(int i=0;i<=n-1;i++)printf("%d\n",ans[i]);

    return 0;
}
```

6.3.2 BZOJ1176 Mokia

```
//二维矩形区域求和
//离线算法，强制在线用kd-tree或树套树
#include<bits/stdc++.h>
```

```cpp
using namespace std;

typedef long long ll;
const int maxn=2e6+5;
const int maxl=2e5+5;

int s,w;
int ans[maxn];
struct BIT{
   int c[maxn];
   inline int lowbit(int x){return x&(-x);}
   inline void add(int x,int d){for(;x<=w;x+=lowbit(x))c[x]+=d;}
   inline int getsum(int x){int r=0;for(;x;x-=lowbit(x))r+=c[x];return r;}
}bit;


int n;
struct node{
   int x,y,val,time;
   int type,id;
}nd[maxl],tmp[maxl];
inline bool cmpxy(node nd1,node nd2){
   if(nd1.x!=nd2.x)return nd1.x<nd2.x;
   if(nd1.y!=nd2.y)return nd1.y<nd2.y;
   return nd1.id<nd2.id;
}

void cdq(int l,int r){
   if(l>=r)return;
   int mid=(l+r)/2;

   for(int i=l;i<=r;i++){
      if(nd[i].time<=mid && nd[i].type==0)
      bit.add(nd[i].y,nd[i].val);
      else if(nd[i].time>=mid+1 && nd[i].type==1){
         if(nd[i].val==1)
         ans[nd[i].id]+=bit.getsum(nd[i].y);
         else
         ans[nd[i].id]-=bit.getsum(nd[i].y);
      }
   }
   for(int i=l;i<=r;i++) if(nd[i].time<=mid && nd[i].type==0)
   bit.add(nd[i].y,-nd[i].val);

   int t1=l-1,t2=mid;
   for(int i=l;i<=r;i++){
      if(nd[i].time<=mid)tmp[++t1]=nd[i];
      else tmp[++t2]=nd[i];
   }
   for(int i=l;i<=r;i++)nd[i]=tmp[i];

   cdq(l,mid);cdq(mid+1,r);
}

int pos[maxn];
int main()
{
   scanf("%d %d",&s,&w);
   int op,x1,y1,x2,y2,a;
   int tot=0;
   while(1){
      scanf("%d",&op);
      if(op==1){
         scanf("%d %d %d",&x1,&y1,&a);
         nd[++n]=(node){x1,y1,a,n,0,0};
```

```
        }
        else if(op==2){
            scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            ans[++tot]=(x2-x1+1)*(y2-y1+1)*s;
            nd[++n]=(node){x2, y2, 1, n, 1, tot};
            nd[++n]=(node){x2, y1-1, -1, n, 1,  tot};
            nd[++n]=(node){x1-1, y2, -1, n, 1,  tot};
            nd[++n]=(node){x1-1, y1-1, 1, n, 1,  tot};
        }
        else break;
    }

    sort(nd+1,nd+1+n,cmpxy);
    cdq(1,n);

    for(int i=1;i<=tot;i++)printf("%d\n",ans[i]);

    return 0;
}
```

6.3.3 优化 1D/1D DP

$$f[i]=\max(f[i],\frac{f[j]Rate[j]}{Rate[j]A[j]+B[j]}A[i]+\frac{f[j]}{Rate[j]A[j]+B[j]}B[i])$$
$$y[j]=\frac{f[j]Rate[j]}{Rate[j]A[j]+B[j]},x[j]=\frac{f[j]}{Rate[j]A[j]+B[j]}$$
$$f[i]=\max_j y[j]A[i]+x[j]B[i]$$
$$\Leftrightarrow y[j]=-\frac{B[i]}{A[i]}x[j]+\frac{f[i]}{A[i]}(y=kx+b)\Leftrightarrow\max_j b$$

```
//BZOJ1492
//cdq分治维护下凸包 (横坐标不要求升序)
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=1e5+5;
const double eps=1e-9;
const double INF=1e100;

struct node{
    int id;
    double x,y,k;
}q[maxn],tmp[maxn];
bool cmp(const node& nd1,const node& nd2){return nd1.k>nd2.k;}
double A[maxn],B[maxn],rate[maxn],dp[maxn];
int n;
node convex[maxn];
double slope(const node& nd1,const node& nd2){
    if(abs(nd2.x-nd1.x)<eps)return INF;
    return (nd2.y-nd1.y)/(nd2.x-nd1.x);
}
void cdq(int l,int r)
{
    if(l==r){
        //l=q[l].id;
        dp[l]=max(dp[l],dp[l-1]);
        q[l].x=dp[l]/(rate[l]*A[l]+B[l]);
        q[l].y=rate[l]*q[l].x;
        return;
    }

    int mid=(l+r)>>1;
    for(int i=l,t1=l,t2=mid+1;i<=r;i++){
        if(q[i].id<=mid)tmp[t1++]=q[i];
        else tmp[t2++]=q[i];
    }
    for(int i=l;i<=r;i++)q[i]=tmp[i];
    cdq(l,mid);

    int sz=0;
    for(int i=l;i<=mid;i++){
        while(sz>=2 && slope(convex[sz-1],convex[sz])<slope(convex[sz],q[i]))--sz;
        convex[++sz]=q[i];
    }
    for(int i=mid+1,j=1;i<=r;i++){
        while(j<sz && slope(convex[j],convex[j+1])>q[i].k)++j;
        int id=q[i].id;
        dp[id]=max(dp[id],A[id]*convex[j].y+B[id]*convex[j].x);
    }

    cdq(mid+1,r);

    int t=l,t1=l,t2=mid+1;
    while(t1<=mid && t2<=r){
        if(q[t1].x<q[t2].x)tmp[t++]=q[t1++];
        else tmp[t++]=q[t2++];
    }
    while(t1<=mid)tmp[t++]=q[t1++];
    while(t2<=r)tmp[t++]=q[t2++];
    for(int i=l;i<=r;i++)q[i]=tmp[i];
}

int main()
{
    scanf("%d%lf",&n,&dp[0]);
    for(int i=1;i<=n;i++){
        scanf("%lf %lf %lf",&A[i],&B[i],&rate[i]);
        q[i].k=-B[i]/A[i]; q[i].id=i;
    }
    sort(q+1,q+1+n,cmp);

    cdq(1,n);
    printf("%.3lf\n",dp[n]);
    return 0;
}
```

6.3.4 Dynamic MST

```
//nlog^2n
//另见LCT + 线段树分治
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 5e4 + 5;
const int INF = 1e9 + 7;

int n, m, q, cnt[18], w[maxn], c[maxn];
ll ans[maxn];
struct Operation {
    int k, d;
} op[maxn];
struct Edge {
    int u, v, w, id;
    bool operator<(const Edge& b) const { return w < b.w; }
```

```
} e[18][maxn], tmp[maxn], s[maxn];

// Union Find Set
int fa[maxn], sz[maxn];
void init(int nn, Edge* a)
{
   for (int i = 1; i <= nn; i++) {
      fa[a[i].u] = a[i].u;
      fa[a[i].v] = a[i].v;
      sz[a[i].u] = sz[a[i].v] = 1;
   }
}
int find(int x) { return fa[x] == x ? x : (fa[x] = find(fa[x])); }
bool merge(int x, int y)
{
   x = find(x), y = find(y);
   if (x == y) return false;
   if (sz[x] > sz[y])
   swap(x, y);
   fa[x] = y;
   sz[y] += sz[x];
   return true;
}

// Dynamic MST
void Contraction(int& nn, ll& val)
{
   int top = 0;
   init(nn, tmp);
   sort(tmp + 1, tmp + 1 + nn);
   for (int i = 1; i <= nn; i++) {
      if (merge(tmp[i].u, tmp[i].v))
      s[++top] = tmp[i];
   }
   init(top, s);
   for (int i = 1; i <= top; i++) {
      if (s[i].w != -INF && merge(s[i].u, s[i].v))
      val += s[i].w;
   }
   top = 0;
   for (int i = 1; i <= nn; i++) {
      if (find(tmp[i].u) != find(tmp[i].v))
      s[++top] = (Edge){ find(tmp[i].u), find(tmp[i].v), tmp[i].w, tmp[i].id };
   }
   for (int i = 1; i <= top; i++)
   c[tmp[i].id] = i, tmp[i] = s[i];
   nn = top;
}
void Reduction(int& nn)
{
   int top = 0;
   init(nn, tmp);
   sort(tmp + 1, tmp + 1 + nn);
   for (int i = 1; i <= nn; i++) {
      if (merge(tmp[i].u, tmp[i].v) || tmp[i].w==INF)
      s[++top] = tmp[i];
   }
   for (int i = 1; i <= top; i++)
   c[tmp[i].id] = i, tmp[i] = s[i];
   nn = top;
}
void cdq(int l, int r, int dep, ll val)
{
   int nn = cnt[dep];
   if (l == r)
```

```
    w[op[l].k] = op[l].d;
    for (int i = 1; i <= nn; i++) {
        e[dep][i].w = w[e[dep][i].id];
        tmp[i] = e[dep][i];
        c[tmp[i].id] = i;
    }
    if (l == r) {
        ans[l] = val;
        init(nn, tmp);
        sort(tmp + 1, tmp + nn + 1);
        for (int i = 1; i <= nn; i++) {
            if (merge(tmp[i].u, tmp[i].v))
            ans[l] += tmp[i].w;
        }
        return;
    }
    for (int i = l; i <= r; i++)
    tmp[c[op[i].k]].w = -INF;
    Contraction(nn, val);
    for (int i = l; i <= r; i++)
    tmp[c[op[i].k]].w = INF;
    Reduction(nn);
    for (int i = 1; i <= nn; i++)
    e[dep + 1][i] = tmp[i];
    cnt[dep + 1] = nn;
    int mid = (l + r) >> 1;
    cdq(l, mid, dep + 1, val);
    cdq(mid + 1, r, dep + 1, val);
}

int main()
{
    scanf("%d %d %d", &n, &m, &q);
    for (int i = 1; i <= m; i++) {
        scanf("%d %d %d", &e[0][i].u, &e[0][i].v, &e[0][i].w);
        e[0][i].id = i, w[i] = e[0][i].w;
    }
    for (int i = 1; i <= q; i++)
    scanf("%d %d", &op[i].k, &op[i].d);
    cnt[0] = m;
    cdq(1, q, 0, 0);
    for (int i = 1; i <= q; i++)
    printf("%lld\n", ans[i]);
    return 0;
}
```

6.4 分块

```
#include <bits/stdc++.h>
#define N 200005
using namespace std;
typedef long long ll;
int n, m;
int belong[N], a[N];
struct node {
    map<int, int> mp;
    int l, r, color;
    int len() {
        return r - l + 1;
    }
} block[1500];
void build() {
    int size = sqrt((double)n + 0.5);
    int cnt = n / size;
    if (n % cnt != 0) cnt++;
    for (int i = 1; i <= cnt; i++) {
        block[i].l = (i - 1) * size + 1;
        block[i].r = i * size;
        block[i].mp.clear();
        block[i].color = -1;
    }
    block[cnt].r = n;
    for (int i = 1; i <= n; i++) {
        belong[i] = (i - 1) / size + 1;
    }
    for (int i = 1; i <= n; i++) {
        block[belong[i]].mp[a[i]]++;
    }
}
void pushdown(int x) {
    if (~block[x].color) {
        for (int i = block[x].l; i <= block[x].r; i++) {
        a[i] = block[x].color;
    }
    block[x].mp.clear();
    block[x].mp[block[x].color] = block[x].len();
    block[x].color = -1;
    }
}
void update(int l, int r, int z) {
    int L = belong[l], R = belong[r];
    for (int i = L + 1; i < R; i++) {
        block[i].color = z;
    }
    if (L != R) {
        pushdown(L); pushdown(R);
        for (int i = l; i <= block[L].r; i++) {
            block[L].mp[a[i]]--;
            block[L].mp[z]++;
            a[i] = z;
        }
        for (int i = block[R].l; i <= r; i++) {
            block[R].mp[a[i]]--;
            block[R].mp[z]++;
            a[i] = z;
        }
    }
    else {
        pushdown(L);
        for (int i = l; i <= r; i++) {
            block[L].mp[a[i]]--;
            block[L].mp[z]++;
            a[i] = z;
        }
    }
}
ll query(int l, int r, int z) {
    ll ans = 0;
    int L = belong[l], R = belong[r];
    for (int i = L + 1; i < R; i++) {
        if (~block[i].color) {//如果这上面有之前的lazy标记，那原mp中颜色作废，WA*4
            if (block[i].color == z)
            ans += block[i].len();
        }
        else if (block[i].mp.count(z) != 0) {
            ans += block[i].mp[z];
        }
    }
    if (L != R) {
        pushdown(L); pushdown(R);
```

```
      for (int i = l; i <= block[L].r; i++) {
      ans += (a[i] == z);
      }
      for (int i = block[R].l; i <= r; i++) {
         ans += (a[i] == z);
      }
   }
   else {
      pushdown(L);
      for (int i = l; i <= r; i++) {
         ans += (a[i] == z);
      }
   }
   return ans;
}
int main() {
   while (scanf("%d%d", &n, &m) != EOF) {
      for (int i = 1; i <= n; i++) {
         scanf("%d", &a[i]);
      }
      build();
      while (m--) {
         int q, l, r, z;
         scanf("%d%d%d%d", &q, &l, &r, &z);
         l++; r++;
         if (q == 1) {
            update(l, r, z);
         }
         else {
            printf("%lld\n", query(l, r, z));
         }
      }
   }
   return 0;
}
```

6.5 k 维偏序

6.5.1 CDQ 分治

```
#include<bits/stdc++.h>

#define Cpy(x,y) memcpy(x,y,sizeof(x))
#define Set(x,y) memset(x,y,sizeof(x))
#define FILE "partial_order"
#define mp make_pair
#define pb push_back
#define RG register
#define il inline
using namespace std;
typedef unsigned long long ull;
typedef vector<int>VI;
typedef long long ll;
typedef double dd;
const dd eps=1e-6;
const int mod=1e9+7;
const int N=50010;
const int M=1e6+10;
const int inf=2147483647;
const ll INF=1e18+1;
const ll P=100000;
namespace IO{
   const int maxn=(1<<21)+1;
   char ibuf[maxn],*iS,*iT,c;int f;
   inline char getc(){
        return iS==iT?(iT=(iS=ibuf)+fread(ibuf,1,maxn,stdin),iS==iT?EOF:*iS++):*iS++;
    }
    template<class T>inline void read(T &x){
        for(f=1,c=getc();(c<'0'||c>'9');c=getc())f=c=='-'?-1:1;
        for(x=0;(c>='0'&&c<='9');c=getc())x=(x<<1)+(x<<3)+(c^48);
        x*=f;
    }
}
using IO::read;
il void file(){
    srand(time(NULL)+rand());
    freopen(FILE".in","r",stdin);
    freopen(FILE".out","w",stdout);
}
const int K=5;//K维偏序
int n,ans;
struct node{int d[K],tg[K];}Q[K][N];
bool cmp(node x,node y,int k){
    if(k==K-2)return x.d[k]<y.d[k];
    return x.d[k]<y.d[k]||(x.d[k]==y.d[k]&&x.d[k+1]==y.d[k+1]);
}
il int pd(node x,int v){
    for(RG int i=1;i<K-2;i++)if(x.tg[i]!=v)return 0;return 1;
}
int t[N];
il void insert(int i){for(i;i<=n;i+=(i&-i))t[i]++;}
il void clear(int i){for(i;i<=n;i+=(i&-i))t[i]=0;}
il int query(int i){int r=0;for(i;i;i-=(i&-i))r+=t[i];return r;}
void cdq(int l,int r,int k){
    if(l==r)return;RG int mid=(l+r)>>1;cdq(l,mid,k);cdq(mid+1,r,k);
    for(RG int i=l,p1=l,p2=mid+1,sum=0;i<=r;i++)
    if(p2>r||(p1<=mid&&cmp(Q[k-1][p1],Q[k-1][p2],k))){
        if(k==K-2&&pd(Q[k-1][p1],0))insert(Q[k-1][p1].d[k+1]);
        Q[k][i]=Q[k-1][p1++];Q[k][i].tg[k]=0;
    }
    else{
        if(k==K-2&&pd(Q[k-1][p2],1))ans+=query(Q[k-1][p2].d[k+1]-1);
        Q[k][i]=Q[k-1][p2++];Q[k][i].tg[k]=1;
    }
    for(RG int i=l;i<=r;i++)Q[k-1][i]=Q[k][i],clear(Q[k][i].d[k+1]);
    if(k!=K-2)cdq(l,r,k+1);
}
int main()
{
    file();read(n);
    for(RG int k=0;k<K;k++)
    for(RG int i=1;i<=n;i++){
        if(k)read(Q[0][i].d[k]);
        else Q[0][i].d[k]=i;
        Q[0][i].tg[k]=-1;
    }
    cdq(1,n,1);
    printf("%d\n",ans);
    return 0;
}
```

6.5.2 bitset+ 分块

```
//O(k*n*sqrt(n))
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=5e4+5;
const int SZ=sqrt(N)+5;
bitset<N>s[6][SZ];
```

```
int n,m,q,score[6];
pair<int,int>a[6][N];
int sz,cnt,belong[N],L[SZ],R[SZ];
void build(){
    sz=sqrt(n);cnt=n/sz;if(n%sz)cnt++;
    for(int i=1;i<=cnt;i++){
        L[i]=(i-1)*sz+1;
        R[i]=min(i*sz,n);
        for(int j=L[i];j<=R[i];j++)belong[j]=i;
    }
    for(int j=1;j<=5;j++)for(int i=1;i<=cnt;i++)s[j][i].reset();
    for(int j=1;j<=5;j++)sort(a[j]+1,a[j]+1+n);
    for(int j=1;j<=5;j++){
        for(int i=1;i<=cnt;i++){
            s[j][i]=s[j][i-1];
            for(int k=L[i];k<=R[i];k++)s[j][i].set(a[j][k].second);
        }
    }
}

int query(){
    bitset<N>ans,tmp;
    ans.set(); ans.reset(0);
    for(int j=1;j<=5;j++){
        int l=1,r=n,mid,id=1;
        while(l<=r){
            mid=(l+r)>>1;
            if(a[j][mid].first<=score[j])id=mid,l=mid+1;
            else r=mid-1;
        }
        int block=id/sz;
        tmp=s[j][block];
        for(int i=L[block+1];i<=n && a[j][i].first<=score[j];i++)tmp.set(a[j][i].second);
        ans&=tmp;
    }
    return ans.count();
}

int main()
{
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%d %d",&n,&m);
        for(int i=1;i<=n;i++){
            for(int j=1;j<=5;j++){
                scanf("%d",&score[j]);
                a[j][i]=make_pair(score[j],i);
            }
        }
        build();
        scanf("%d",&q);
        int lastans=0;
        while(q--){
            for(int j=1;j<=5;j++){
                scanf("%d",&score[j]);
                score[j]^=lastans;
            }
            lastans=query();
            printf("%d\n",lastans);
        }
    };
    return 0;
}
```

6.6 莫队

6.6.1 带修改

```
#include<bits/stdc++.h>
#define FOR(i,l,r) for(int (i)=l;(i)<=(r);(i)++)
#define pb push_back
#define fi first
#define se second
using namespace std;
const int maxn=1e6+5,sz=400;
int n,m,k;
int l=1,r=0,now=0;
char ch[5];
struct node {
    int l,r,id,time;
}Q[maxn];
struct node1{
    int p,from,to;
}R[maxn];
int pos[maxn];
int cnt[maxn],s[maxn];

inline bool cmp(const node& A,const node&B) {
    if(pos[A.l]!=pos[B.l]) return pos[A.l]<pos[B.l];
    if(pos[A.r]!=pos[B.r]) return pos[A.r]<pos[B.r];
    return A.time<B.time;
}

int Ans[maxn],ans;
void Ins(int x) {
    cnt[x]++;
    if(cnt[x]==1)ans++;
}
void Del(int x) {
    cnt[x]--;
    if(cnt[x]==0)ans--;
}
void in_time(int x){
    int pp=R[x].p,to=R[x].to,&from=R[x].from;
    if(pp>=l && pp<=r)Del(s[pp]);
    from=s[pp];
    s[pp]=to;
    if(pp>=l && pp<=r)Ins(s[pp]);
}
void out_time(int x){
    int pp=R[x].p,to=R[x].to,from=R[x].from;
    if(pp>=l && pp<=r)Del(s[pp]);
    s[pp]=from;
    if(pp>=l && pp<=r)Ins(s[pp]);
}
int main() {
    memset(Ans,-1,sizeof(Ans));
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++){
        scanf("%d",&s[i]);
    }
    int tim=0,q=0,x,y;
    for(int i=1;i<=m;++i) {
        scanf("%s",ch);
        scanf("%d%d",&x,&y);
        if(ch[0]=='R'){
            R[++tim]=(node1){x,0,y};
        }
        else if(ch[0]=='Q'){
            Q[++q]=(node){x,y,i,tim};
```

```
      }
   }
   for(int i=1;i<=n;++i)pos[i]=i/sz;
   sort(Q+1,Q+q+1,cmp);
   for(int i=1;i<=q;++i){
      while(r<Q[i].r)++r,Ins(s[r]);
      while(l<Q[i].l)Del(s[l]),++l;
      while(l>Q[i].l)--l,Ins(s[l]);
      while(r>Q[i].r)Del(s[r]),--r;

      while(now<Q[i].time)++now,in_time(now);
      while(now>Q[i].time)out_time(now),--now;
      Ans[Q[i].id]=ans;
   }
   for(int i=1;i<=m;++i)if(Ans[i]!=Ans[0])printf("%d\n",Ans[i]);
   return 0;
}
```

6.6.2 不带修改

```
#include<bits/stdc++.h>
#define FOR(i,l,r) for(int (i)=l;(i)<=(r);(i)++)
#define pb push_back
#define fi first
#define se second
using namespace std;
const int maxn=1e6+5,sz=400;
int n,m,k;
struct node {
   int l,r,id;
}Q[maxn];
int pos[maxn];
int cnt[maxn],s[maxn];

inline bool cmp(const node& A,const node&B) {
   if(pos[A.l]==pos[B.l]) {
      return pos[A.l]&1?A.r<B.r:A.r>B.r;
   }
   return pos[A.l]<pos[B.l];
}

int Ans[maxn],ans;
void Ins(int x) {
   cnt[x]=1;
   if(cnt[x+1]+cnt[x-1]==0)ans++;
   else if(cnt[x+1]+cnt[x-1]==2)ans--;
}
void Del(int x) {
   cnt[x]=0;
   if(cnt[x-1]+cnt[x+1]==0)ans--;
   else if(cnt[x-1]+cnt[x+1]==2)ans++;
}
int main() {
   int t;
   scanf("%d",&t);
   while(t--){
      ans=0;
      memset(cnt,0,sizeof(cnt));
      scanf("%d %d",&n,&m);
      for(int i=1;i<=n;i++){
         scanf("%d",&s[i]);
      }
      for(int i=1;i<=m;++i) {
         scanf("%d %d",&Q[i].l,&Q[i].r);
         Q[i].id=i;
```

```
        }
        for(int i=1;i<=n;++i)pos[i]=i/sz;
        sort(Q+1,Q+1+m,cmp);
        int l=1,r=0;
        for(int i=1;i<=m;++i){
            while(r<Q[i].r)++r,Ins(s[r]);
            while(l<Q[i].l)Del(s[l]),++l;
            while(l>Q[i].l)--l,Ins(s[l]);
            while(r>Q[i].r)Del(s[r]),--r;
            Ans[Q[i].id]=ans;
        }
        for(int i=1;i<=m;++i)printf("%d\n",Ans[i]);
    }
    return 0;
}
```

6.6.3 回滚莫队

```
只加不减
1. 对原序列进行分块，并对询问按照如下的方式排序：以左端点所在的块升序为第一关键字，以右端点升序为第二关键字
2. 对于处理所有左端点在块T内的询问，先将莫队区间左端点初始化为R[T]+1，右端点初始化为R[T]，这是一个空区间
3. 对于左右端点在同一个块中的询问，直接暴力扫描回答即可。
4. 对于左右端点不在同一个块中的所有询问，由于其右端点升序，对右端点只做加点操作，总共最多加点n次
5. 对于左右端点不在同一个块中的所有询问，其左端点是可能乱序的，每一次从R[T]+1的位置出发，只做加点操作，到达询问位置即可，
   每一个询问最多加√n次。回答完询问后，撤销本次移动左端点的所有改动，使左端点回到R[T]+1的位置
6.复杂度O(n√n)

只减不加
1. 对原序列进行分块，并对询问按照如下的方式排序：以左端点所在的块升序为第一关键字，以右端点降序序为第二关键字
2. 对于处理所有左端点在块T内的询问，先将莫队区间左端点初始化为L[T]，右端点初始化为n，这是一个大区间
3. 对于左右端点在同一个块中的询问，直接暴力扫描回答即可。
4. 对于左右端点不在同一个块中的所有询问，由于其右端点降序，从n的位置开始，对右端点只做删点操作，总共最多删点n次
5. 对于左右端点不在同一个块中的所有询问，其左端点是可能乱序的，每一次从L[T]的位置出发，只做删点操作，到达询问位置即可，每一
   个询问最多加√n次。回答完询问后，撤销本次移动左端点的所有改动，使左端点回到L[T]的位置
(6).复杂度O(n√n)
```

```
//区间max{val*出现次数}
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=1e5+5;
int n,q;
int x[maxn],t[maxn],m;

struct Query{
    int l,r,id;
}Q[maxn];
int pos[maxn],L[maxn],R[maxn],sz,tot;
int cnt[maxn],__cnt[maxn];
ll ans[maxn];

inline bool cmp(const Query& A,const Query& B){
    if(pos[A.l]==pos[B.l])return A.r<B.r;
    return pos[A.l]<pos[B.l];
}

inline void Add(int v,ll &Ans){
    ++cnt[v];
    Ans=max(Ans,1LL*cnt[v]*t[v]);
}
inline void Del(int v){
    --cnt[v];
}
```

```
int main()
{
    scanf("%d %d",&n,&q);
    for(int i=1;i<=n;i++)scanf("%d",&x[i]),t[++m]=x[i];
    for(int i=1;i<=q;i++)scanf("%d %d",&Q[i].l,&Q[i].r),Q[i].id=i;
    sz=sqrt(n); tot=n/sz;
    for(int i=1;i<=tot;i++){
        L[i]=(i-1)*sz+1;
        R[i]=i*sz;
    }
    if(R[tot]<n){++tot;L[tot]=R[tot-1]+1;R[tot]=n;}
    for(int i=1;i<=tot;i++)for(int j=L[i];j<=R[i];j++)pos[j]=i;
    sort(Q+1,Q+1+q,cmp);

    sort(t+1,t+1+m);
    m=unique(t+1,t+1+m)-(t+1);
    for(int i=1;i<=n;i++)x[i]=lower_bound(t+1,t+1+m,x[i])-t;

    int l=1,r=0,last_block=0,__l;
    ll Ans=0,tmp;
    for(int i=1;i<=q;i++){
        if(pos[Q[i].l]==pos[Q[i].r]){
            for(int j=Q[i].l;j<=Q[i].r;j++)++__cnt[x[j]];
            for(int j=Q[i].l;j<=Q[i].r;j++)ans[Q[i].id]=max(ans[Q[i].id],1LL*t[x[j]]*__cnt[x[j]]);
            for(int j=Q[i].l;j<=Q[i].r;j++)--__cnt[x[j]];
            continue;
        }
        if(pos[Q[i].l]!=last_block){
            while(r>R[pos[Q[i].l]])Del(x[r]),--r;
            while(l<R[pos[Q[i].l]]+1)Del(x[l]),++l;
            Ans=0; last_block=pos[Q[i].l];
        }
        while(r<Q[i].r)++r,Add(x[r],Ans);
        __l=l; tmp=Ans;
        while(__l > Q[i].l)--__l,Add(x[__l],tmp);
        ans[Q[i].id]=tmp;
        //roll back
        while(__l<l)Del(x[__l]),++__l;
    }
    for(int i=1;i<=q;i++)printf("%lld\n",ans[i]);
    return 0;
}
```

```
//区间mex(最小未出现的自然数)
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=2e5+5;
int n,q;
int x[maxn];

struct Query{
    int l,r,id;
}Q[maxn];
int pos[maxn],L[maxn],R[maxn],sz,tot;
int cnt[maxn],__cnt[maxn];
int ans[maxn];

inline bool cmp(const Query& A,const Query& B){
    if(pos[A.l]==pos[B.l])return A.r>B.r;
    return pos[A.l]<pos[B.l];
}

inline void Add(int v){
    if(v>n+1)return;
    ++cnt[v];
}
inline void Del(int v,int &Ans){
    if(v>n+1)return;
    --cnt[v];
    if(cnt[v]==0)Ans=min(Ans,v);
}

int main()
{
    scanf("%d %d",&n,&q);
    for(int i=1;i<=n;i++)scanf("%d",&x[i]);
    for(int i=1;i<=q;i++)scanf("%d %d",&Q[i].l,&Q[i].r),Q[i].id=i;
    sz=sqrt(n); tot=n/sz;
    for(int i=1;i<=tot;i++){
        L[i]=(i-1)*sz+1;
        R[i]=i*sz;
    }
    if(R[tot]<n){++tot;L[tot]=R[tot-1]+1;R[tot]=n;}
    for(int i=1;i<=tot;i++)for(int j=L[i];j<=R[i];j++)pos[j]=i;
    sort(Q+1,Q+1+q,cmp);

    int l=1,r=0,last_block=0,__l;
    int Ans,tmp,__Ans=0;

    for(int j=1;j<=n;j++)if(x[j]<=n+1)++__cnt[x[j]];
    while(__cnt[__Ans])++__Ans;
    for(int j=1;j<=n;j++)if(x[j]<=n+1)--__cnt[x[j]];

    for(int i=1;i<=q;i++){
        if(pos[Q[i].l]==pos[Q[i].r]){
            for(int j=Q[i].l;j<=Q[i].r;j++)if(x[j]<=n+1)++__cnt[x[j]];
            while(__cnt[ans[Q[i].id]])++ans[Q[i].id];
            for(int j=Q[i].l;j<=Q[i].r;j++)if(x[j]<=n+1)--__cnt[x[j]];
            continue;
        }
        if(pos[Q[i].l]!=last_block){
            while(r<n)++r,Add(x[r]);
            while(l<L[pos[Q[i].l]])Del(x[l],__Ans),++l;
            Ans=__Ans; last_block=pos[Q[i].l];
        }
        while(r>Q[i].r)Del(x[r],Ans),--r;
        __l=l; tmp=Ans;
        while(__l < Q[i].l)Del(x[__l],tmp),++__l;
        ans[Q[i].id]=tmp;

        //roll back
        while(__l>l)--__l,Add(x[__l]);
    }
    for(int i=1;i<=q;i++)printf("%d\n",ans[i]);
    return 0;
}
```

6.7 离散化

```
sort(b+1,b+1+n);
int cnt=unique(b+1,b+n+1)-b-1;
for(int i=1;i<=n;i++)id[i]=lower_bound(b+1,b+1+n,a[i])-b;
```

6.8 RMQ

```
int mi[maxn][20],lg[maxn];
```

```
void initRMQ()
{
    memset(mi,0x3f,sizeof(mi));
    lg[1]=0;
    for(int i=2;i<=n;i++)lg[i]=lg[i>>1]+1;
    for(int i=1;i<=n;i++)mi[i][0]=height[i];
    for(int j=1;j<=lg[n];j++)
        for(int i=1;i<=n;i++)
            mi[i][j]=min(mi[i][j-1],mi[i+(1<<j-1)][j-1]);
}

int query(int l,int r)
{
    int d=lg[r-l+1];
    return min(mi[l][d],mi[r-(1<<d)+1][d]);;
}
```

6.9 单调栈

```
//单调栈
l[1]=1;
for(int i=2;i<=n;i++){
    int x=i;
    while(x>1 && a[i]<=a[x-1])x=l[x-1];
    l[i]=x;
}

r[n]=n;
for(int i=n-1;i>=1;i--){
    int x=i;
    while(x<n && a[i]<=a[x+1])x=r[x+1];
    r[i]=x;
}
```

```
//单调双端队列
int h1=0,h2=0,t1=1,t2=1,p=0;
for(int k=1;k<=n;k++){//right col
    while(h1>=t1 && ma[k]>=ma[q1[h1]])h1--;
    q1[++h1]=k;
    while(h2>=t2 && mi[k]<=mi[q2[h2]])h2--;
    q2[++h2]=k;
    while(h1>=t1 && h2>=t2 && ma[q1[t1]]-mi[q2[t2]]>m){
        p++;
        if(h1>=t1 && q1[t1]<=p)t1++;
        if(h2>=t2 && q2[t2]<=p)t2++;
    }
    ans=max(ans,(j-i+1)*(k-p));
}
```

```
//RMQ优化单调栈
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int maxn = 202020;
const int inf = 0x3f3f3f3f;
int h[maxn];
int l[maxn][19],r[maxn][19],s[maxn],top;
ll sl[maxn][19],sr[maxn][19];
int main(void) {
    //freopen("j.in","r",stdin);
    int n; scanf("%d",&n);
    h[0]=h[n]=inf;
    for (int i=1;i<n;i++) scanf("%d",&h[i]);
    s[top++]=0;
    for (int i=1;i<n;i++) {
        for (;h[i]>=h[s[top-1]];) top--;
        l[i][0]=s[top-1];
        sl[i][0]=(i-s[top-1])*(ll)h[i];
        for (int j=1;j<=18;j++) {
            l[i][j]=l[l[i][j-1]][j-1];
            sl[i][j]=sl[i][j-1]+sl[l[i][j-1]][j-1];
        }
        s[top++]=i;
    }
    for (int j=0;j<=18;j++) r[n][j]=n;
    top=0;
    s[top++]=n;
    for (int i=n-1;i>=1;i--) {
        for (;h[i]>=h[s[top-1]];) top--;
        r[i][0]=s[top-1];
        sr[i][0]=(s[top-1]-i)*(ll)h[i];
        for (int j=1;j<=18;j++) {
            r[i][j]=r[r[i][j-1]][j-1];
            sr[i][j]=sr[i][j-1]+sr[r[i][j-1]][j-1];
        }
        s[top++]=i;
    }
    int q; scanf("%d",&q);
    for (int i=1;i<=q;i++) {
        int a,b; scanf("%d%d",&a,&b);
        ll ans=0;
        if (a<b) {
            int now = b-1;
            for (int t=18;t>=0;t--) {
                if (l[now][t] >= a) {
                    ans += sl[now][t];
                    now = l[now][t];
                }
            }
            ans += sl[now][0];
        } else {
            int now = b;
            for (int t=18;t>=0;t--) {
                if (r[now][t] <= a-1) {
                    ans += sr[now][t];
                    now = r[now][t];
                }
            }
            ans += sr[now][0];
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

6.10 希伯特坐标转换

```
void rot(int n, Point pt, int rx, int ry) {
    if (ry == 0) {
        if (rx == 1) {
            pt.x = n - 1 - pt.x;
            pt.y = n - 1 - pt.y;
        }
        swap(x,y);
    }
}
```

```
//Hilbert代码到XY坐标
void d2xy(int n, int d, Point pt) {
    int rx, ry, s, t = d;
    pt.x = pt.y = 0;
    for (s = 1; s < n; s *= 2) {
        rx = 1 & (t / 2);
        ry = 1 & (t ^ rx);
        rot(s, pt, rx, ry);
        pt.x += s * rx;
        pt.y += s * ry;
        t /= 4;
    }
}

//XY坐标到Hilbert代码转换
int xy2d(int n, Point pt) {
    int rx, ry, s, d = 0;
    for (s = n / 2; s > 0; s /= 2) {
        rx = ((pt.x & s) > 0) ? 1 : 0;
        ry = ((pt.y & s) > 0) ? 1 : 0;
        d += s * s * ((3 * rx) ^ ry);
        rot(s, pt, rx, ry);
    }
    return d;
}
```

6.11 曼哈顿坐标系和切比雪夫坐标系转换

$$\begin{gathered}\text{曼哈顿坐标系转切比雪夫坐标系:}(x,y)=(x+y,x-y)\\ \text{切比雪夫坐标系转曼哈顿坐标系:}(x,y)=(\frac{x+y}{2},\frac{x-y}{2})\end{gathered} \tag{7}$$

6.12 动态连通性问题

6.12.1 离线可撤销并查集 + 线段树分治

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
const int maxn=5010;
const int maxm=500010;

struct Edge{int x,y,l,r;}e[maxm];
int tot;

struct Query{int x,y;};

int n,m;
int tim[maxn][maxn],ans[maxm];

namespace Union_Find_Set{
    int fa[maxn],dep[maxn],p1[maxn],p2[maxn],top;
    void init(int n){
        top=0;
        for(int i=1;i<=n;i++){
            fa[i]=i;
            dep[i]=1;
        }
    }
    int find(int x){return fa[x]==x?x:find(fa[x]);}
    void merge(int x,int y){
        int fx=find(x),fy=find(y);
        if(fx==fy)return;
        if(dep[fx]<=dep[fy]){
            fa[fx]=fy;
            top++; p1[top]=fx; p2[top]=dep[fy];
            dep[fy]=max(dep[fy],dep[fx]+1);
        }
        else{
            fa[fy]=fx;
            top++; p1[top]=fy; p2[top]=dep[fx];
            dep[fx]=max(dep[fx],dep[fy]+1);
        }
    }
    void del(int t){
        while(top>t){
            dep[fa[p1[top]]]=p2[top];
            fa[p1[top]]=p1[top];
            top--;
        }
    }
}using namespace Union_Find_Set;

namespace Segment_Tree{
    vector<int>edge[maxm<<2];
    void insert(int x,int l,int r,int L,int R,int id){
        if(l==L && r==R){edge[x].push_back(id);return;}
        int mid=(l+r)>>1;
        if(R<=mid)insert(x<<1,l,mid,L,R,id);
        else if(L>mid)insert(x<<1|1,mid+1,r,L,R,id);
        else
        {
            insert(x<<1,l,mid,L,mid,id);
            insert(x<<1|1,mid+1,r,mid+1,R,id);
        }
    }
}using namespace Segment_Tree;

Query q[maxm];
void solve(int x,int l,int r){
    int now=top;
    for(int i=0;i<(int)edge[x].size();i++)merge(e[edge[x][i]].x,e[edge[x][i]].y);
    if(l==r){
        if(q[l].x)puts((find(q[l].x)==find(q[l].y))?"Y":"N");
    }
    else{
        int mid=(l+r)>>1;
        solve(x<<1,l,mid); solve(x<<1|1,mid+1,r);
    }
    del(now);
}

int main()
{
    scanf("%d %d",&n,&m);
    int op,x,y;
    for(int i=1;i<=m;i++){
        scanf("%d %d %d",&op,&x,&y);
        switch(op){
            case 0:
            tim[x][y]=++tot;
            e[tot].x=x; e[tot].y=y; e[tot].l=i; e[tot].r=m;
            break;
            case 1:
            e[tim[x][y]].r=i;
            break;
            case 2:
            q[i].x=x; q[i].y=y;
            break;
        }
    }
    init(n);
    for(int i=1;i<=tot;i++)insert(1,1,m,e[i].l,e[i].r,i);
    solve(1,1,m);
    return 0;
}
```

6.12.2 在线 ETT+ 分层图

```
/*
我们维护log(点数)个图G_i以及这些图的生成森林F_i,每个图G_{i+1}都是G{i}删去一些边形成的,我们给每条边附一个权值level表示让
    这条边出现的最大G_i.
当我们插入一条边, 就在G_0与F_0中加入那条边, 它的level为0.
我们询问时只要询问F_0中是否联通.
当我们删除一条边(u--v:level), 就删除i<=level所有的G_i中的这条边, 然后我们考虑,在如果这次删除将F_i中劈成Sub(u)子树与Sub(
    v)子树, 我们需要寻找一条替代这条边的边.假设|Sub(u)|<=|Sub(v)|,那么我们在Sub(u)处于G_i中的出边遍历,寻找一条Sub(u)->
    Sub(v)的边.如果找到了那就是一条替代边,对于遍历到不符合条件的边我们知道它可以插入G_{i+1}中,且level++.
这样每条边最多被遍历到log次,因为每次增加level都会使它所在的最大F_i大小至少减半,而它只会一次担任替代边.
*/
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXL = 16, N = 5005;
char buf[1 << 20], *p1, *p2;
#define GC (p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 1 << 20, stdin), p1 == p2) ? 0 : *p1++)
inline int read() {
    char t = GC;
    int x;
    while (t < 48 || t > 57) t = GC;
    for (x = 0; t > 47 && t < 58; t = GC) x = (x << 3) + (x << 1) + (t ^ 48);
    return x;
}
inline int Rand() {
    static unsigned int seed = 998244353;
    return seed = (seed ^ 1313131311) * 1234321237 + 19260817;
}
int n, m;
struct Graph {
    set<int> mt[N];
    void add(int x, int y) {
        mt[x].insert(y);
        mt[y].insert(x);
    }
    void del(int x, int y) {
        mt[x].erase(y);
        mt[y].erase(x);
    }
    bool find(int x, int y) { return mt[x].count(y); }
    bool empty(int x) { return mt[x].empty(); }
} G[MAXL];
struct node {
    node *ls, *rs, *fa;
    bool hav;
    int pid, tid, sz, w;
} pool[10000000], *bin[10000000], *null;
int tl;
void Init() {
    null = pool;
    null->ls = null->rs = null->fa = null;
    for (int i = 1; i < N * MAXL; i++) bin[i] = pool + i;
```

```
}
struct ETT {
   Graph *G;
   int tote;
   set<int> e[N];
   map<int, node *> mt[N];
   node *id[N];
   node *newnode(int id, int tid) {
      node *x = bin[++tl];
      x->pid = id;
      x->tid = tid;
      x->hav = id && !G->empty(id);
      x->sz = 1;
      x->ls = x->rs = x->fa = null;
      return x;
   }
   void delnode(node *x) { bin[tl--] = x; }
   void pushup(node *x) {
      x->hav = (x->pid && !G->empty(x->pid)) || x->ls->hav || x->rs->hav;
      x->sz = x->ls->sz + x->rs->sz + 1;
   }
   void splitup(node *x, node *&l, node *&r) {
      if (x->fa == null)
      return;
      node *y = x->fa;
      x->fa = null;
      if (y->ls == x) {
         y->ls = r;
         r->fa = y;
         pushup(y);
         r = y;
      } else {
         y->rs = l;
         l->fa = y;
         pushup(y);
         l = y;
      }
      splitup(y, l, r);
   }
   void split(node *x, node *&l, node *&r) {
      l = x->ls;
      r = x->rs;
      x->ls = x->rs = l->fa = r->fa = null;
      pushup(x);
      splitup(x, l, r);
   }
   node *merge(node *l, node *r) {
      if (l == null)
      return r;
      if (r == null)
      return l;
      if (Rand() & 16) {
         l->rs = merge(l->rs, r);
         l->rs->fa = l;
         return pushup(l), l;
      } else {
         r->ls = merge(l, r->ls);
         r->ls->fa = r;
         return pushup(r), r;
      }
   }
   void makeroot(node *&x) {
      if (x == null)
      return;
      node *l, *r;
```

```
        split(x, l, r);
        x = merge(merge(x, r), l);
    }
    node *findrt(node *x) {
        while (x->fa != null) x = x->fa;
        while (x->ls != null) x = x->ls;
        return x;
    }
    node *findrt(int x) { return findrt(id[x]); }
    void add(int u, int v, bool lev) {
        node *x = id[u], *y = id[v];
        if (x != null && y != null && findrt(x) == findrt(y)) {
            pushup(x);
            pushup(y);
            while (x->fa != null) x = x->fa, pushup(x);
            while (y->fa != null) y = y->fa, pushup(y);
            return;
        }
        makeroot(x);
        makeroot(y);
        mt[u][v] = newnode(x == null ? u : 0, u);
        mt[v][u] = newnode(y == null ? v : 0, v);
        if (lev)
        e[u].insert(v), e[v].insert(u);
        merge(merge(merge(mt[u][v], y), mt[v][u]), x);
        if (x == null)
        id[u] = mt[u][v];
        if (y == null)
        id[v] = mt[v][u];
    }
    void getid(int x) {
        if (mt[x].empty()) {
            id[x] = null;
            return;
        }
        id[x] = mt[x].begin()->second;
        node *l, *r;
        split(id[x], l, r);
        id[x]->pid = x;
        pushup(id[x]);
        merge(merge(id[x], r), l);
    }
    int del(int u, int v) {
        node *x = mt[u][v], *y = mt[v][u], *l, *r;
        mt[u].erase(v);
        mt[v].erase(u);
        split(x, l, r);
        merge(r, l);
        split(y, l, r);
        int t0 = l->sz, t1 = r->sz;
        if (x->pid)
        getid(u);
        if (y->pid)
        getid(v);
        delnode(x);
        delnode(y);
        return t1 < t0 ? u : v;
    }
    void init(int x) {
        tote = 1;
        fill(id, id + n + 1, null);
        G = ::G + x;
    }
} T[MAXL];
namespace D_Graph {
```

```
void addlevel(int level, node *x) {
   if (x == null || !x->hav)
   return;
   if (x->pid) {
      int u = x->pid;
      for (auto v : T[level].e[u]) {
         if (u >= v)
         continue;
         G[level].del(u, v);
         G[level + 1].add(u, v);
         T[level + 1].add(u, v, 1);
      }
      T[level].e[u].clear();
   }
   addlevel(level, x->ls);
   addlevel(level, x->rs);
   T[level].pushup(x);
}
node *xrt;
int X, Y;
bool findin_G(int level, int x) {
   while (!G[level].empty(x)) {
      int u = *G[level].mt[x].begin();
      if (T[level].findrt(u) != xrt)
      return X = x, Y = u, 1;
      G[level].del(x, u);
      G[level + 1].add(x, u);
      T[level + 1].add(x, u, 0);
   }
   return 0;
}
bool findin_T(int level, node *x) {
   if (x == null || !x->hav)
   return 0;
   if (x->pid)
   if (findin_G(level, x->pid))
   return 1;
   if (findin_T(level, x->ls))
   return 1;
   if (findin_T(level, x->rs))
   return 1;
   x->hav = 0;
   return 0;
}
void find_replacement(int level, int x) {
   node *p = T[level].id[x];
   xrt = p;
   if (p == null)
   findin_G(level, x);
   else
   T[level].makeroot(p), addlevel(level, p), findin_T(level, p);
}
void add(int x, int y) {
   G[0].add(x, y);
   T[0].add(x, y, 1);
}
void del(int x, int y) {
   for (int i = MAXL - 1; i >= 0; i--) {
      if (!G[i].find(x, y))
      continue;
      G[i].del(x, y);
      if (!T[i].mt[x].count(y))
      return;
      T[i].e[x].erase(y);
      T[i].e[y].erase(x);
```

```
          X = Y = 0;
          for (int j = i; j >= 0; j--) {
            int t = T[j].del(x, y);
            if (!X) {
              find_replacement(j, t);
              if (X)
              T[j].add(X, Y, 1);
            } else
            T[j].add(X, Y, 0);
          }
          return;
        }
    }
    bool isconnected(int x, int y) {
      return T[0].id[x] != null && T[0].id[y] != null && T[0].findrt(x) == T[0].findrt(y);
    }
} // namespace D_Graph
int main() {
    int i, opt, x, y, ans = 0;
    n = read();
    m = read();
    Init();
    for (i = 0; i < MAXL; i++) T[i].init(i);
    for (i = 1; i <= m; i++) {
      opt = read();
      x = read() ^ ans;
      y = read() ^ ans;
      if (opt == 0)
      D_Graph::add(x, y);
      else if (opt == 1)
      D_Graph::del(x, y);
      else {
        ans = D_Graph::isconnected(x, y);
        puts(ans ? "Y" : "N");
        ans = ans ? x : y;
      }
    }
    return 0;
}
```

6.13 随机算法求解 n 皇后

```
//QS2 algorithm, get a solution of n queen in seconds
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

const double C1 = 0.45;
const double C2 = 32;

void initQueen(int N , vector <int> &queen) {
    vector <int> array(N);
    for(int i = 0; i < N ; i++) array[i] = i;
    queen.clear();
    while(N) {
      double t = rand()%N;
      queen.push_back(array[t]);
      array[t] = array[--N];
    }
}

ll compute_collisions(const vector<int> &queen,vector<int> &dn,vector<int> &dp) {
    for(int t =0 ; t < 2*(int)queen.size() ; t++) dn[t] = dp[t] = 0;

    for(int i = 0 ; i < (int)queen.size() ; i++) {
      int j = queen.at(i);
      dn[i+j]++;
      dp[queen.size()-1+i-j]++;
   }

   ll collisions = 0;
   for(int k =0 ; k < 2*(int)queen.size()-1; k++) {
      collisions +=dn[k]*(dn[k]-1)/2;
      collisions +=dp[k]*(dp[k]-1)/2;
   }
   return collisions;
}

int compute_attacks(const vector<int> &queen,vector<int> &dn,vector<int> &dp,vector<int>&attack) {
   attack.clear();
   int attackedLines = 0;
   for(int i =0 ; i<(int)queen.size(); i++) {
      int j = queen.at(i);
      if(dn.at(i+j) > 1 || dp.at(queen.size()-1+i-j) > 1) {
         attack.push_back(i);
         attackedLines++;
      }
   }
   return attackedLines;
}

int random(int k ,int size) {
   int val =rand()%size;
   while(val == k)
   val = rand()%size;
   return val;
}


int calc_collisions(const vector<int> &queen,vector<int> &dn,vector<int> &dp,int a,int b) {
   int i,j,counts = 0 ;

   i = queen[a];
   j = queen[b];

   counts -= dn[a+i] - 1;
   counts -= dn[b+j] - 1;
   if(a+i == b+j) counts++;

   counts -= dp[queen.size()-1+a-i] - 1;
   counts -= dp[queen.size()-1+b-j] - 1;
   if(a-i == b-j) counts++;

   counts += dn[a+j] ;
   counts += dn[b+i] ;
   if(a+j == b+i) counts++;

   counts += dp[queen.size()-1+a-j] ;
   counts += dp[queen.size()-1+b-i] ;
   if(a-j == b-i) counts++;

   return counts;
}

void updateQueen(vector<int> &queen,vector<int> &dn,vector<int> &dp,int a,int b) {
   int i = queen.at(a);
   int j = queen.at(b);

   dn[a+i]--;
   dn[b+j]--;
```

```
    dp[queen.size()-1+a-i]--;
    dp[queen.size()-1+b-j]--;

    int t = queen[a];
    queen[a] = queen[b];
    queen[b] = t;

    dn[a+j]++;
    dn[b+i]++;
    dp[queen.size()-1+a-j]++;
    dp[queen.size()-1+b-i]++;
}

void printQueen(vector<int>&queen) {
    for(int i = 0 ; i < (int)queen.size(); i++) {
        for(int j = 0; j < (int)queen.size(); j++) {
            if(j == queen.at(i))
            cout<<"Q";
            else cout<<".";
        }
        cout<<endl;
    }
}

void n_queen(int N) {
    ll collisions = N;
    int num_of_attack;
    int swapCount = 0;

    vector<int> queen;//皇后所在位置
    vector<int> attack;//冲突行

    vector<int> dn(2*N);//次对角线
    vector<int> dp(2*N);//主对角线

    if(N == 2||N == 3) {
        cout<<"无解"<<endl;
        return ;
    }

    while(collisions) {
        initQueen(N,queen);
        collisions = compute_collisions(queen, dn, dp);
        num_of_attack = compute_attacks(queen, dn, dp, attack);

        int limit = C1*collisions;
        int loopSteps = 0;
        while(loopSteps < C2*N) {

            for(int k = 0 ; k < num_of_attack ; k++) {
                int i = attack.at(k);
                int j = random(i,N);

                int c;
                if(( c = calc_collisions(queen,dn,dp,i,j)) < 0) { //if swap_ok
                    //perform_swap
                    collisions += c;
                    updateQueen(queen,dn,dp,i,j);
                    swapCount++;

                    if(collisions == 0)break;
                    if(collisions < limit) {
                        limit = C1*collisions;
                        num_of_attack = compute_attacks(queen, dn, dp, attack);
                    }
            }
         }
         if(collisions == 0)break;
         loopSteps +=num_of_attack;
      }
   }
   cout<<"swap time(s): "<<swapCount<<endl;
   if(N<=80)printQueen(queen);
}

int main() {
   int N=100000;
   clock_t startTime,endTime;

   cin>>N;

   srand(time(0));

   startTime = clock();
   n_queen(N);
   endTime = clock();

   double totalTime = 1.0*(endTime - startTime)/CLOCKS_PER_SEC;
   cout<<"cost: "<<totalTime<<"秒"<<endl;

   return 0;
}
```

6.14 rope

```
//文艺平衡树（区间翻转）
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

#include <ext/rope>
using namespace __gnu_cxx;
/*
常用成员函数:
1.push_back(x)
2.在pos处插入: insert(pos,x)
3.从pos开始删除x个元素: erase(pos,x)
4.从pos开始换成x: replace(pos,x)
5.从pos开始的len个元素换为x:copy(pos,len,x)
6.从pos开始提取x个元素: substr(pos,x)
7.访问第pos个元素:at(pos)/[pos]
8.将串s从pos开始的n个元素连接到当前串的结尾:append(s,pos,n)
*/
rope<int> T1, T2, T;
int n, m;
int main()
{
   scanf("%d %d", &n, &m);
   for (int i = 1; i <= n; ++i)
   T1.push_back(i), T2.push_back(n - i + 1);
   int l, r;
   for (int i = 1; i <= m; ++i) {
      scanf("%d %d", &l, &r);
      --l, --r;
      T = T1.substr(l, r - l + 1);
      T1 = T1.substr(0, l) + T2.substr(n - r - 1, r - l + 1) + T1.substr(r + 1, n - r - 1);
      T2 = T2.substr(0, n - r - 1) + T + T2.substr(n - l, l);
   }
   for (int i = 0; i < n; ++i)
   printf("%d ", T1[i]);
}
```

6.15 pbds

6.15.1 平衡树

```
#include <bits/stdc++.h>

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

using namespace std;
typedef long long ll;
typedef ll key_type;
typedef null_mapped_type value_type;
typedef tree<key_type, value_type, less<key_type>, rb_tree_tag, tree_order_statistics_node_update> rbtree;
/*
参数意义:
1.key_type表示树中key的类型, 如int
2.value_type表示树中value的类型, null_type表示无映射
若G++版本较老, null_type可能需要换为null_mapped_type
3.排序方式, less表示从小到大, greater标志从大到小
4.表示树的类型, rb_tree_tag表示红黑树, splay_tree_tag表示伸展树
5.节点更新策略, 填tree_order_statistics_node_update则可以询问rank和查第k大
常用成员函数:
1.插入: insert()
2.删除: erase()
3.询问K的rank: order_of_key()
4.询问第K小: find_by_order()
5.lower_bound()
6.upper_bound()
7.合并两颗树: a.join(b), 要求两棵树的key值不相交, 即无重复元素
8.将一颗树按key值划分问两棵树: a.split(v,b),小于等于v的属于a,其余属于b
复杂度全为O(logn)
*/
rbtree T;
int n;
int main()
{
    scanf("%d", &n);
    ll op, val;
    for (int i = 1; i <= n; i++) {
        scanf("%lld %lld", &op, &val);
        switch (op) {
            case 1:
            T.insert((val << 20) + i);
            break;
            case 2:
            T.erase(T.lower_bound(val << 20));
            break;
            case 3:
            printf("%d\n", T.order_of_key(val << 20) + 1);
            break;
            case 4:
            printf("%lld\n", (*T.find_by_order(val - 1)) >> 20);
            break;
            case 5:
            printf("%lld\n", (*--T.lower_bound(val << 20)) >> 20);
            break;
            case 6:
            printf("%lld\n", (*T.lower_bound((val + 1) << 20)) >> 20);
            break;
        }
    }
}
```

6.15.2 可并堆

```
//pb_ds的优先队列优化dijstra
#include<bits/stdc++.h>
#include<ext/pb_ds/priority_queue.hpp>
using namespace std;
using namespace __gnu_pbds;
typedef long long ll;
typedef pair<ll,int> pi;
typedef __gnu_pbds::priority_queue<pi,greater<pi>,pairing_heap_tag > heap;
const int maxn=1e6+5,maxm=1e7+5;
const ll INF=(((1ll<<62)-1)<<1)+1;
int n,m,cnt,last[maxn];
int T,rxa,rxc,rya,ryc,rp;
heap::point_iterator id[maxn];
int x,y,z;
ll dis[maxn];
struct Edge{int to,next,v;}e[maxm];
void addedge(int u,int v,int w){
    e[++cnt].to=v;e[cnt].next=last[u];last[u]=cnt;e[cnt].v=w;
}
void dijkstra(){
    heap q;
    for(int i=1;i<=n;i++)dis[i]=INF;
    dis[1]=0;id[1]=q.push(make_pair(0,1));
    while(!q.empty()){
        int now=q.top().second;q.pop();
        for(int i=last[now];i;i=e[i].next)
        if(e[i].v+dis[now]<dis[e[i].to]){
            dis[e[i].to]=e[i].v+dis[now];
            if(id[e[i].to]!=0)
            q.modify(id[e[i].to],make_pair(dis[e[i].to],e[i].to));
            else id[e[i].to]=q.push(make_pair(dis[e[i].to],e[i].to));
        }
    }
}
int main()
{
    scanf("%d %d",&n,&m);
    scanf("%d %d %d %d %d %d",&T,&rxa,&rxc,&rya,&ryc,&rp);
    int a,b;
    for(int i=1;i<=T;i++){
        x=((ll)x*rxa+rxc)%rp;
        y=((ll)y*rya+ryc)%rp;
        a=min(x%n+1,y%n+1);
        b=max(y%n+1,y%n+1);
        addedge(a,b,100000000-100*a);
    }
    for(int i=1;i<=m-T;i++){
        scanf("%d %d %d",&x,&y,&z);
        addedge(x,y,z);
    }
    dijkstra();
    printf("%lld",dis[n]);
    return 0;
}
/*
建议使用tag:pairing_heap_tag,binomial_heap_tag
push()插入，返回point_iterator
modify(point_iterator it,const_reference r_new_val)修改对应点的值
erase(point_iterator it)删除对应点
join(priority &other)合并当前堆和堆other，other会被清空
pop(),top(),size(),empty()等同std::priority_queue
```

```
*/
```