目录

头文件

```cpp
#include <sstream>
#include <fstream>
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <cstring>
#include <stack>
#include <queue>
#include <cmath>
#include <ctime>
#include <utility>
#include <cassert>
#include <bitset>
using namespace std;
#define REP(I,N) for (I=0;I<N;I++)
#define rREP(I,N) for (I=N-1;I>=0;I--)
#define rep(I,S,N) for (I=S;I<N;I++)
#define rrep(I,S,N) for (I=N-1;I>=S;I--)
#define FOR(I,S,N) for (I=S;I<=N;I++)
#define rFOR(I,S,N) for (I=N;I>=S;I--)

#define DEBUG
#ifdef DEBUG
#define debug(...) fprintf(stderr, __VA_ARGS__)
#define deputs(str) fprintf(stderr, "%s\n",str)
#else
#define debug(...)
#define deputs(str)
#endif // DEBUG
typedef unsigned long long ULL;
typedef unsigned long long ull;
typedef unsigned int ui;
typedef long long LL;
typedef long long ll;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
const int INF=0x3f3f3f3f;
const LL INFF=0x3f3f3f3f3f3f3f3fll;
const LL M=1e9+7;
const LL maxn=1e6+107;
const double pi=acos(-1.0);
const double eps=0.0000000001;
LL gcd(LL a, LL b) {return b?gcd(b,a%b):a;}
template<typename T>inline void pr2(T x,int k=64) {ll i; REP(i,k) debug("%d",(x>>i)&1); putchar(' ');}
template<typename T>inline void add_(T &A,int B,ll MOD=M) {A+=B; (A>=MOD) &&(A-=MOD);}
template<typename T>inline void mul_(T &A,ll B,ll MOD=M) {A=(A*B)%MOD;}
template<typename T>inline void mod_(T &A,ll MOD=M) {A%=MOD; A+=MOD; A%=MOD;}
template<typename T>inline void max_(T &A,T B) {(A<B) &&(A=B);}
template<typename T>inline void min_(T &A,T B) {(A>B) &&(A=B);}
template<typename T>inline T abs(T a) {return a>0?a:-a;}
inline ll powMM(ll a, ll b, ll mod=M) {
    ll ret=1;
    for (; b; b>>=1ll,a=a*a%mod)
        if (b&1) ret=ret*a%mod;
    return ret;
}
int startTime;
void startTimer() {startTime=clock();}
void printTimer() {debug("/--- Time: %ld milliseconds ---/\n",clock()-startTime);}
```

杂物

首先是没啥用的两个板子

```
void msort(int le,int ri) {//逆序对
    if (le==ri) return;
    int mid=(le+ri)>>1,i=le,j=mid+1,k=i;
    msort(le,mid); msort(j,ri);
    while (i<=mid||j<=ri) {
        if (i==mid+1) {b[k++]=a[j++]; ans+=mid-i+1;}
        else if (j==ri+1) b[k++]=a[i++];
        else if (a[i]<=a[j]) b[k++]=a[i++];
        else {b[k++]=a[j++]; ans+=mid-i+1;}
    }
    for (i=le; i<=ri; i++) a[i]=b[i];
}

void fqsort(int l,int r) {//O(n)第k大数
    int le=l,ri=r,m;
    m=a[le];
    while (le<ri) {
        while (le<ri&&a[ri]<=m) ri--; a[le]=a[ri];
        while (le<ri&&a[le]>=m) le++; a[ri]=a[le];
    } if (le==k) printf("%d\n",m);
    else if (le>k) fqsort(l,le-1);
    else fqsort(le+1,r);
}
```

并查集(维护链)

```
inline int getfa(int x){
    if (fa[x]==x) return x;
    int y=getfa(fa[x]);
    if (fa[x]!=y) sum[x]+=sum[fa[x]];
    fa[x]=y;
    return y;
}
```

读入挂

```
namespace fastIO {//感觉没问题，测试几次
#define BUF_SIZE 100000
    namespace Istream {
        bool IOerror = 0;
        inline char ic() {
            static char
buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=buf+BUF_SIZE;
            if (p1==pend) {
                p1=buf;
                pend=buf+fread(buf,1,BUF_SIZE,stdin);
                if (pend == p1) {IOerror = 1; return -1;}
            } return *p1++;
        }
        inline bool blank(char ch) {
            return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';
        }
        template<typename T>
        inline void readPositive(T &x) {//no
            char ch; x=0;
            while (blank(ch=ic()));
            if (IOerror) return;
            for (x=0; '0'<=ch&&ch<='9'; ch=ic())
x=x*10+ch-'0';
        }
        template<typename T>
        inline void read(T &x) {
            char ch; T op=1; x=0;
            while (blank(ch=ic()));
            if (IOerror) return;
            if (ch=='-') op=-1,ch=ic();
            for (x=0; '0'<=ch&&ch<='9'; ch=ic())
x=x*10+ch-'0';
            x*=op;
        }
        inline void read(char &c) {
            c=ic();
        }
        inline void read(char *s) { //len
            char ch;
            while (blank(ch=ic()));
            if (IOerror) return;
        for (; !blank(ch)&&!IOerror; ch=ic()) *s++=ch;
        *s='\0';
    }
}
namespace Ostream {
    char buf[BUF_SIZE], *p1 = buf, *pend = buf +
BUF_SIZE;
    inline void flush() {
        fwrite(buf,1,p1-buf,stdout);
        p1=buf;
    }
    inline void oc(char ch) {
        if (p1 == pend) flush();
        *p1++=ch;
    }
    inline void println() {
        oc('\n');
    }
    template<typename T>
    inline void print(T x) {
        static char s[27],*s1=s;
        if (!x) *s1++='0';
        if (x<0) oc('-'),x=-x;
        while (x) *s1++=x%10+'0',x/=10;
        do {s1--; oc(*s1);} while (s1!=s);
    }
    inline void print(char s) {
        oc(s);
    }
    inline void print(char *s) {
        for (; *s; oc(*s++));
    }
    inline void print(const char *s) {
        for (; *s; oc(*s++));
    }
    inline void print(string s) {
        for (unsigned i=0; i<s.length(); i++) oc(s[i]);
    }
    struct _flush {
        ~_flush() {flush();}
    } fflush;
};
template<typename T>
inline void read(T &x) {Istream::readPositive(x);}
inline void read(char *x) {Istream::read(x);}
    template<typename T>
    inline void print(T x) {Ostream::print(x);}
    template<typename T>
    inline void println(T x) {print(x); Ostream::oc('\n');}
}
using namespace fastIO;
```

其他挂

扩栈挂

```
#ifdef OPENSTACK
    int size = 256 << 20; // 256MB
    char *p = (char*)malloc(size) + size;
    #if (defined _WIN64) or (defined __unix)
        __asm__("movq %0, %%rsp\n" :: "r"(p));
    #else
        __asm__("movl %0, %%esp\n" :: "r"(p));
    #endif
#endif
```

注意最后加 exit(0);

玄学挂

```
#pragma comment(linker, "/stack:200000000")
#pragma GCC optimize("Ofast,no-stack-protector")
#pragma GCC
target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=nativ
e")
```

然后加上并行计算(计组)

```
#pragma GCC optimize("Ofast,no-stack-protector")
#pragma GCC target("avx")
```

平板电视

1、红黑树

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_cxx;
using namespace __gnu_pbds;
typedef
tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_
update> rbtree;

/*
定义一颗红黑树
```

```
int 关键字类型
null_type 无映射(低版本 g++为 null_mapped_type)
less<int>从小到大排序
rb_tree_tag 红黑树（splay_tree_tag）
tree_order_statistics_node_update 结点更新
插入 t.insert();
删除 t.erase();
Rank:t.order_of_key();
第 K 值:t.find_by_order();
前驱:t.lower_bound();
后继 t.upper_bound();
a.join(b)b 并入 a  前提是两棵树的 key 的取值范围不相交
a.split(v,b)key 小于等于 v 的元素属于 a，其余的属于 b
T.lower_bound(x)    >=x 的 min 的迭代器
T.upper_bound((x)   >x 的 min 的迭代器
T.find_by_order(k)  有 k 个数比它小的数
*/

rbtree T;
rbtree::iterator it;
```

2、 Rope

```
using namespace std;
using namespace __gnu_cxx;
/*
1）运算符：rope 支持 operator += -= + - < ==
2）输入输出：可以用<<运算符由输入输出流读入或输出。
3）长度/大小：调用 length()，size()都可以哦
4）插入/添加等：
append(const string&)
substr(start,length)
push_back(x);//在末尾添加 x
insert(pos,x);//在 pos 插入 x，自然支持整个 char 数组的一次插入
erase(pos,x);//从 pos 开始删除 x 个
copy(pos,len,x);//从 pos 开始到 pos+len 为止用 x 代替
replace(pos,x);//从 pos 开始换成 x
substr(pos,x);//提取 pos 开始 x 个
at(x)/[x];//访问第 x 个元素
*/
rope<int> V;
```

3、 二项堆(这里是 dijkstra)

```
#include<iostream>
#include<cstdio>
#include<cstring>
    #include<ext/pb_ds/priority_queue.hpp>
    #define ll long long
    #define pa pair<ll,int>
    #define llinf 9000000000000000000LL
    using namespace std;
    using namespace __gnu_pbds;
    typedef
__gnu_pbds::priority_queue<pa,greater<pa>,pairing_heap_tag >
heap;
    int n,m,cnt,last[1000005];
    int T,rxa,rxc,rya,ryc,rp;
    heap::point_iterator id[1000005];
    int x,y,z;
    ll dis[1000005];
    struct data {int to,next,v;} e[10000005];
    inline int read() {
        int x=0,f=1; char ch=getchar();
        while (ch<'0'||ch>'9') {if (ch=='-')f=-1; ch=getchar();}
        while (ch>='0'&&ch<='9') {x=x*10+ch-'0'; ch=getchar();}
        return x*f;
    }
    void insert(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]=llinf;
        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));
                }
        }
    }
```

Dancing Links

1、不可重复

```
//dlx:求解精确覆盖
//link 的意思是,r 覆盖了 c
//暴力枚举,n 个点覆盖 m 个格子; 注意一定要 init
struct DLX {
    const static int maxn=1e5+7;
    const static int maxd=1e4+7;
    int n,m,size;
    int
U[maxn],D[maxn],R[maxn],L[maxn],col[maxn],row[maxn];
    int H[maxd],S[maxd];//S:cnt
    int ans[maxn];
    void init(int _n,int _m) {
        n=_n; m=_m; int i;
        FOR(i,0,m) {
            S[i]=0;
            U[i]=D[i]=i;
            L[i]=i-1,R[i]=i+1;
        } R[m]=0; L[0]=m;
        size=m;
        FOR(i,0,n) H[i]=-1;
    }
    void link(int r,int c) {
        S[col[++size]=c]++; row[size]=r;
        D[size]=D[c]; U[D[c]]=size;
        D[c]=size; U[size]=c;
        if (H[r]<0) H[r]=L[size]=R[size]=size;
        else {
            R[size]=R[H[r]];
            L[R[H[r]]]=size;
            L[size]=H[r];
            R[H[r]]=size;
        }
    }
    void remove(int c) {
        L[R[c]]=L[c]; R[L[c]]=R[c];
        for (int i=D[c]; i!=c; i=D[i])
            for (int j=R[i]; j!=i; j=R[j])
                U[D[j]]=U[j],D[U[j]]=D[j],S[col[j]]--;
    }
    void resume(int c) {
        for (int i=U[c]; i!=c; i=U[i])
            for (int j=L[i]; j!=i; j=L[j])
                U[D[j]]=D[U[j]]=j,S[col[j]]++;
        L[R[c]]=R[L[c]]=c;
    }
    //这里是找可行解; 最优解无法加估价函数剪枝
    char g[maxn];
    bool dance(int pos) {
        if (R[0]==0) {
            int i,j;
            REP(i,pos) g[(ans[i]-1)/16]=(ans[i]-1)%16+'A';
            REP(i,16) {REP(j,16) putchar(g[i*16+j]);
puts("");}
            return 1;
        }
        // cnt,pos:选择的个数; ans:选择的值(列)
        // if (pos>=cnt&&cnt!=INF) return;
        // if (R[0]==0) {cnt=min(cnt,pos); return;}
        int c=R[0];
        for (int i=R[0]; i; i=R[i])
            if (S[i]<S[c]) c=i;
        remove(c);
        for (int i=D[c]; i!=c; i=D[i]) {
            ans[pos]=row[i];
            for (int j=R[i]; j!=i; j=R[j]) remove(col[j]);
            if (dance(pos+1)) return 1;
            for (int j=L[i]; j!=i; j=L[j]) resume(col[j]);
        } resume(c);
        return 0;
    }
} dlx;

char g[27][27];
int n,m;
void add(int x,int y,int k) {
    int r=(x*16+y)*16+k;
    dlx.link(r,16*16*0+x*16+y+1);//position
    dlx.link(r,16*16*1+x*16+k);//行
    dlx.link(r,16*16*2+y*16+k);//列
    dlx.link(r,16*16*3+(x/4*4+y/4)*16+k);//块
}
int main() {
    int i,j,k;
    while (~scanf("%s",g[0])) {
        rep(i,1,16) scanf("%s",g[i]);
        dlx.init(16*16*16,16*16*4);
        REP(i,16) REP(j,16) FOR(k,1,16) {
            if (g[i][j]=='-'||g[i][j]=='A'-1+k)
                add(i,j,k);
        }
        static int x=0;
        if (x) puts(""); else x=1;
        dlx.dance(0);
    }
}
```

2、 **可重复**

```
//dlx:求解精确覆盖
//link 的意思是,r 覆盖了 c
//暴力枚举,n 个点覆盖 m 个格子; 注意一定要 init
struct DLX {
    const static int maxn=1e5+7;
    const static int maxd=1e4+7;
    int n,m,size;
    int
U[maxn],D[maxn],R[maxn],L[maxn],col[maxn],row[maxn];
    int H[maxd],S[maxd];//S:cnt
    int ans[maxn];
    void init(int _n,int _m) {
        n=_n; m=_m; int i;
        FOR(i,0,m) {
            S[i]=0;
            U[i]=D[i]=i;
            L[i]=i-1,R[i]=i+1;
        } R[m]=0; L[0]=m;
        size=m;
        FOR(i,0,n) H[i]=-1;
    }
    void link(int r,int c) {
        S[col[++size]=c]++; row[size]=r;
        D[size]=D[c]; U[D[c]]=size;
        D[c]=size; U[size]=c;
        if (H[r]<0) H[r]=L[size]=R[size]=size;
        else {
            R[size]=R[H[r]];
            L[R[H[r]]]=size;
            L[size]=H[r];
            R[H[r]]=size;
        }
    }
    void remove(int c) {
        for (int i=D[c]; i!=c; i=D[i])
            L[R[i]]=L[i],R[L[i]]=R[i];
    }
    void resume(int c) {
        for (int i=U[c]; i!=c; i=U[i])
            L[R[i]]=R[L[i]]=i;
    }
    bool v[maxd];
    //估价函数,函数返回的是至少还需要多少行才能完成重复覆盖
    //如果 max 的话可以直接 cnt{R[]},也就是最多个数
    int f() {
        int ret=0;
        for (int c=R[0]; c; c=R[c]) v[c]=1;
        for (int c=R[0]; c; c=R[c]) {
            if (v[c]) {
                ret++; v[c]=0;
                for (int i=D[c]; i!=c; i=D[i])
                    for (int j=R[i]; j!=i; j=R[j])
                        v[col[j]]=0;
            }
        }
        return ret;
    }
    int cnt;
    void dance(int pos) {
        if (R[0]==0) {cnt=min(cnt,pos); return;}
        if (pos+f()>=cnt) return;
        int c=R[0];
        for (int i=R[0]; i; i=R[i])
            if (S[i]<S[c]) c=i;
        for (int i=D[c]; i!=c; i=D[i]) {
            ans[pos]=row[i];
            remove(i);
            for (int j=R[i]; j!=i; j=R[j]) remove(j);
            dance(pos+1);
            for (int j=L[i]; j!=i; j=L[j]) resume(j);
            resume(i);
        }
    }
} dlx;
int n,m;
int check(int x,int y,int a,int b,double d) {
    return (x-a)*(x-a)+(y-b)*(y-b)<d*d;
}
int x1[maxn],x2[maxn],y1[maxn],y2[maxn];
int main() {
    int T;
    scanf("%d",&T);
    while (T--) {
        int k,i;
        scanf("%d%d%d",&n,&m,&k);
        FOR(i,1,n) scanf("%d%d",&x1[i],&y1[i]);
        FOR(i,1,m) scanf("%d%d",&x2[i],&y2[i]);
        double l=0,r=1500;
        while (r-l>1e-7) {
            int i,j;
            double mid=(l+r)/2;
//              printf("%f %f %f\n",l,mid,r);
            dlx.init(m,n);
            FOR(i,1,n) FOR(j,1,m) {
                if (check(x1[i],y1[i],x2[j],y2[j],mid))
                    dlx.link(j,i);
            }
            dlx.cnt=k+1;
            dlx.dance(0);
            if (dlx.cnt>k) l=mid;
            else r=mid;
        } printf("%.6f\n",l);
    }
}
```

快速乘法(就那个 long double 的)

```
return ( x * y - ( long long ) ( x / ( long double ) MOD * y + 
1e-8 ) * MOD + MOD ) % MOD ;
```

快速乘法(mod 加速的)

```
typedef unsigned long long u64;
typedef __int128_t i128;
typedef __uint128_t u128;
int _,k;
u64 A0,A1,M0,M1,C,M;

struct Mod64 {
    Mod64():n_(0) {}
    Mod64(u64 n):n_(init(n)) {}
    static u64 init(u64 w) { return reduce(u128(w) * r2); }
    static void set_mod(u64 m) {
        mod=m; assert(mod&1);
        inv=m; rep(i,0,5) inv*=2-inv*m;
        r2=-u128(m)%m;
    }
    static u64 reduce(u128 x) {
        u64 
y=u64(x>>64)-u64((u128(u64(x)*inv)*mod)>>64);
        return ll(y)<0?y+mod:y;
    }
    Mod64& operator += (Mod64 rhs) { n_+=rhs.n_-mod; if 
(ll(n_)<0) n_+=mod; return *this; }
    Mod64 operator + (Mod64 rhs) const { return 
Mod64(*this)+=rhs; }
    Mod64& operator -= (Mod64 rhs) { n_-=rhs.n_; if 
(ll(n_)<0) n_+=mod; return *this; }
    Mod64 operator - (Mod64 rhs) const { return 
Mod64(*this)-=rhs; }
    Mod64& operator *= (Mod64 rhs) 
{ n_=reduce(u128(n_)*rhs.n_); return *this; }
    Mod64 operator * (Mod64 rhs) const { return 
Mod64(*this)*=rhs; }
    u64 get() const { return reduce(n_); }
    static u64 mod,inv,r2;
    u64 n_;
};
u64 Mod64::mod,Mod64::inv,Mod64::r2;

u64 pmod(u64 a,u64 b,u64 p) {
    u64 d=(u64)floor(a*(long double)b/p+0.5);
    ll ret=a*b-d*p;
    if (ret<0) ret+=p;
    return ret;
}


void bruteforce() {
    u64 ans=1;
    for (int i=0;i<=k;i++) {
        ans=pmod(ans,A0,M);
        u64 A2=pmod(M0,A1,M)+pmod(M1,A0,M)+C;
        while (A2>=M) A2-=M;
        A0=A1; A1=A2;
    }
    printf("%llu\n",ans);
}

int main() {
    for (scanf("%d",&_);_;_--) {

scanf("%llu%llu%llu%llu%llu%llu%d",&A0,&A1,&M0,&M1,&C,&M,
&k);
            Mod64::set_mod(M);
            Mod64
a0(A0),a1(A1),m0(M0),m1(M1),c(C),ans(1),a2(0);
            for (int i=0;i<=k;i++) {
                ans=ans*a0;
                a2=m0*a1+m1*a0+c;
                a0=a1; a1=a2;
            }
            printf("%llu\n",ans.get());
        }
    }
```

极小 mex 区间

```
//题意是个极小mex区间取得最值，求个什么东西
int n;
struct frac {
    int x,y;//x/y
    frac(int _x=1,int _y=1):x(_x),y(_y) {}
    bool operator < (const frac &A) const {
        return (ll)x*A.y<(ll)y*A.x;
    }
}
frac query(int x) {
} MIN[maxn]; //len
inline int lowbit(int x) {
    return x&-x;
}
void update(int x,frac val) {
    for (; x<=n; x+=lowbit(x))
        MIN[x]=min(val,MIN[x]);
    frac ans;
    for (; x; x-=lowbit(x))
        ans=min(ans,MIN[x]);
    return ans;
}
vector<pair<int,frac> > segment[maxn];
map<int,int> MP;//value,pos
int R[maxn],cnt[maxn];
int A[maxn],pos[maxn];
vector<pair<int,int> > queries[maxn];
frac ans[maxn];
int main() {
    int i,q;
    scanf("%d%d",&n,&q);
    FOR(i,1,n) {
        scanf("%d",&A[i]);
        assert(0<=A[i]&&A[i]<=n);
    } FOR(i,0,n) pos[i]=n+1;
    rFOR(i,1,n) {
        R[i]=pos[A[i]]; pos[A[i]]=i;
    } int now=0,tot=0;
    FOR(i,1,n) {//f[1]
        cnt[A[i]]++;
        if (cnt[now]) {
            while (cnt[now]) now++;
            MP[now]=i;//pos
        }
    } MP[n+1]=n+1;//为了简化操作
    FOR(i,1,n) { //remove this 的贡献
        int position=i;//should_add
        while (1) {//不能直接remove
            auto it=MP.lower_bound(A[i]);
            if (it==MP.end()) break;
            if (it->second>=R[i]) break;//del this; position
            int len=it->second-i+1,x=it->first;//should+
            if (position==i) position=it->second;//first_change

segment[i].push_back(make_pair(it->second,frac(len+1-x,len+1+x)
)); tot++;
            auto itt=it; itt++;
            int _R=itt->second; MP.erase(it);
            if (_R>R[i]) MP[x]=R[i];//insert_more
        }//not!
        if (position!=i&&A[i]) MP[A[i]]=position;
    } assert(tot<=3*n);//最大3n以内; 应该说2n以内
    FOR(i,1,q) {//为了用树状数组倒着查emmmm
        int l,r;
        scanf("%d%d",&l,&r);
        assert(1<=l&&l<=r&&r<=n);
        queries[l].push_back(make_pair(r,i));
    } rFOR(i,1,n) {
        for (auto now:segment[i])
            update(now.first,now.second);
        for (auto now:queries[i])
            ans[now.second]=query(now.first);
    } FOR(i,1,q) {
        int g=gcd(ans[i].x,ans[i].y);
        printf("%d/%d\n",ans[i].x/g,ans[i].y/g);
        }
    }
```

set 暴力修改区间

```
    //UVALive 8191 区间 same
    set<pair<pii,int> > POS;
    int cnt[maxn];
    void update(int col,int x){
        cnt[col]+=x;
    }

    void update(int l,int r,int x){
        auto final=make_pair(make_pair(l,r),x);
        while (l<=r){
            auto
it=POS.upper_bound(make_pair(make_pair(l,INF),0)); it--;
            auto now=*it; POS.erase(it);
            int nxtl=now.first.second+1;
            if (now.first.first<l){
                pair<int,int> remain;
                remain.first=now.first.first;
                remain.second=l-1;
                if (remain.first<=remain.second)
                    POS.insert(make_pair(remain,now.second));
            }
            if (now.first.second>r){
                pair<int,int> remain;
                remain.first=r+1;
                remain.second=now.first.second;
                if (remain.first<=remain.second)
                    POS.insert(make_pair(remain,now.second));
                nxtl=r+1;
            }
            update(now.second,-(nxtl-l));
            update(x,nxtl-l);
            l=nxtl;
        } POS.insert(final);
    }
    int main() {
        int l,c;
        while(~scanf("%d%d%d",&l,&c,&n)){
            int i; POS.clear();
            FOR(i,1,c) cnt[i]=0; cnt[1]=l; int ans=0;
            POS.insert(make_pair(make_pair(1,l),1));
            FOR(i,1,n){
                int p,x,a,b;
                scanf("%d%d%d%d",&p,&x,&a,&b);
                int S=cnt[p];
                int m1=(a+(ll)S*S)%l+1;
                int m2=(a+(ll)(S+b)*(S+b))%l+1;
                if (m1>m2) swap(m1,m2);
                update(m1,m2,x);
            } FOR(i,1,c) ans=max(ans,cnt[i]);
            printf("%d\n",ans);
        }
    }
```

螺旋数列 value

```
    //123
    //894
    //765
    inline ll getValue(ll n,ll x,ll y) {
        ll r = 0;
        x=n-x+1; y=n-y+1;
        if (x<=y && x+y <= n+1) {
            r = x;
            return  4*(r-1)*n - 4*(r-1)*(r-1) +1 + y-r;
        }
        if (x<=y && x+y >= n+1) {
            r = n- y + 1;
            return 4*(r-1)*n - 4*(r-1)*(r-1) + 1 + n-2*r + 1 +
x - r;
        }
        if (x>=y && x+y >= n+1) {
            r = n - x +1;
            return 4*(r-1)*n - 4*(r-1)*(r-1) + 1 + 3*n-6*r + 3
- y + r;
        }
        if (x>=y && x+y <= n+1) {
            r = y;
            return 4*(r-1)*n - 4*(r-1)*(r-1) + 1 + 4*n-8*r + 4
- x + r;
        }
        assert(0);
        return -1;
    }
```

DP

决策单调性优化

```
//决策单调性优化可以处理所有斜率优化的题
//题意:sum{A[l]->A[k],{1<=l<r<=n,k 是 l->r 的路径上最近的标记点}}
//做法:DP; 注意有时 DP[0]甚至 DP[1]都要预处理的
//注意先写好 DP 方程
//注意 DP 方程上代表的意义!
//注意不能转移的地方!一定 continue,否则可能破坏可以优化的性质
//我的理解:从左往右来看,如果 l++,那么切的点只会向右移动,xl,xr 是指转折点可能出现的位置;
//CDQ 分治,传递下去了解可能存在的区间
//每次更新的是 mid 节点
//bfs,dfs 均可,时间均为 log(莫队不影响,莫队时间可证明 nlogn)
//CF868F 题意:切区间 k 段,每段数字出现个数 sigma{n(n-1)/2}最小的个数
LL L1[maxn],L2[maxn],R1[maxn],R2[maxn];//前缀和之和,小技巧
LL getL(int l,int r) { //一个求 l->r 的点到 l 的 sum 和
    return (L2[r]-L2[l])-L1[l]*(r-l);
}
LL getR(int l,int r) {
    return (R2[l]-R2[r])-R1[r]*(r-l);
}

LL pre[maxn],dp[maxn];
struct node {int l,r,xl,xr;};
LL cnt,sum,sum_sum;
queue<node> Q;
void changel(LL val,int seg) {
    sum_sum+=sum*seg*2;
    sum_sum-=cnt*val*seg*2;
    cnt+=seg; sum+=val*seg;
}
void changer(LL val,int seg) {
    sum_sum-=sum*seg*2;
    sum_sum+=cnt*val*seg*2;
    cnt+=seg; sum+=val*seg;
}
int _l,_r;
LL A[maxn];
void changeto(int l,int r) {
    while (_r<r) _r++,changer(A[_r],1);
    while (_l>l) _l--,changel(A[_l],1);
    while (_l<l) changel(A[_l],-1),_l++;
    while (_r>r) changer(A[_r],-1),_r--;
}
void solve(int n) {
    int i;
    Q.push(node{1,n,0,n-1});
    while (Q.size()) {
        auto F=Q.front(); Q.pop();
        int l=F.l,r=F.r,L=F.xl,R=F.xr;//l,r,check_l,check_r
        int m=(l+r)/2,M=L;
        LL &now=dp[m];
        FOR(i,L,min(m-1,R)) {
            //这里 changeto 不会改变复杂度
            LL msum=(m-i)*getL(m,n);
            LL rsum=(n-m+1)*(getR(i+1,m)+i*(A[m]-A[i]));
            if (now>pre[i]-msum-rsum)
                now=pre[i]-msum-rsum,M=i;
        }
        if (l<m) Q.push(node{l,m-1,L,M});
        if (r>m) Q.push(node{m+1,r,M,R});
    }
}
//DP[i]:i_chosen; contains [i]->[i]; [i]->R(i+1->n)
//update:m [i-m]->[i], [i-m]->[m-n] [i-m]->[i-m]
int T;
int n,m,k;
int i,j;
int main() {
    while (~scanf("%d%d",&n,&k)) {
        FOR(i,1,n) scanf("%lld",&A[i]);
        A[0]=A[1]; A[n+1]=A[n];
        FOR(i,1,n) L1[i]=A[i]-A[i-1]+L1[i-1];
        FOR(i,1,n) L2[i]=L2[i-1]+L1[i];

        rFOR(i,1,n) R1[i]=A[i+1]-A[i]+R1[i+1];
        rFOR(i,1,n) R2[i]=R2[i+1]+R1[i];
```

```
            _l=1; _r=0; sum=sum_sum=cnt=0;
            changeto(1,n);
            FOR(i,0,n) dp[i]=sum_sum;
            FOR(i,1,k) {
                int i;
                FOR(i,0,n) pre[i]=dp[i];
                solve(n);
//              FOR(m,1,n) FOR(i,0,m-1){
////                    changeto(i+1,m);
////                    cal:-=[m,n]->[i](differ)+[i+1-m](to m)
////                    cal:-=[i+1,m]->[m,n](to m)
//                  LL msum=(m-i)*getL(m,n);
//                  LL
rsum=(n-m+1)*(getR(i+1,m)+i*(A[m]-A[i]));
//                  dp[m]=min(dp[m],pre[i]-msum-rsum);
//              }
            }
            LL ans=dp[0];
            FOR(i,1,n) ans=min(ans,dp[i]);
            printf("%lld\n",ans);
        }
    }
```

斜率优化

```
//HDU 3480//斜率优化
//题意:一堆数字,切成 k 份,每块的代价为(max-min)^2
//dp 方程:dp[i][j]=min{dp[k][j-1]+(a[i]-a[k+1])^2};
//dp 方程:
//dp[i][j]=min{dp[k][j-1]+a[k+1]^2-2*a[i]*a[k+1]}+a[i]^2
//k=(dp[k][j-1](pre)+a[k+1]^2)/(a[k+1]),常数 2*a[i]
//斜率优化本质是维护一个下凸壳
int n,m,i,j,k,t;
int a[maxn],pre[maxn],dp[maxn];
int head,tail;
int Q[maxn];//id
inline int getY(int id){
    return pre[id]+a[id+1]*a[id+1];
}
inline int getX(int id){
    return a[id+1];
}
int main(){
    int T,X=0;
    scanf("%d",&T);
    while (T--){
        scanf("%d%d",&n,&m);
        FOR(i,1,n) scanf("%d",&a[i]);
        sort(a+1,a+1+n);
        int qi,qj,qk;
        FOR(i,1,n) dp[i]=(a[i]-a[1])*(a[i]-a[1]);
        FOR(j,2,m){
            FOR(i,1,n) pre[i]=dp[i];
            head=tail=0;
            dp[0]=0;Q[tail++]=0;
            FOR(i,1,n){
                while (head+1<tail){
                    qi=Q[head],qj=Q[head+1];
                    if
(getY(qj)-getY(qi)<=2*a[i]*(getX(qj)-getX(qi))) head++;
                    else break;
                }qi=Q[head];
                dp[i]=pre[qi]+(a[i]-a[qi+1])*(a[i]-a[qi+1]);
                while (head+1<tail){
                    qi=Q[tail-2];qj=Q[tail-1];qk=i;
                    int
y1=getY(qj)-getY(qi),x1=getX(qj)-getX(qi);
                    int
y2=getY(qk)-getY(qj),x2=getX(qk)-getX(qj);
                    if (y2*x1<=y1*x2) tail--;//y2/x2>y1/x1
                    else break;
                }Q[tail++]=i;
            }
        }
        printf("Case %d: %d\n",++X,dp[n]);
    }
}
```

四边形不等式优化

```
//HDU 3516//四边形不等式优化
//题意:给定一个从左上往右下的图，只能往下往右连，
求一个构造使得所有的边长度总和最小
//dp 方程:
//dp[i][j]=max{dp[i][k]+dp[k+1][j]+x[k+1]-x[i]+y[k]-
y[j]};
//能用：满足:
//w[i][j]+w[i'][j']<=w[i][j']+w[i'][j];
//w[i'][j']<=w[i][j],那么决策区间包含
struct node{
    int x,y;
}a[maxn];
int n,m,i,j,k,t;
int dp[maxn][maxn],pos[maxn][maxn];
int main(){
    while (~scanf("%d",&n)){
        FOR(i,1,n) scanf("%d%d",&a[i].x,&a[i].y),pos[i][i]=i;
        FOR(i,1,n) FOR(j,i+1,n) dp[i][j]=INF;
        FOR(t,1,n-1){
            FOR(i,1,n-t){
                j=i+t;
                FOR(k,pos[i][j-1],min(j-1,pos[i+1][j])){
                    int
now=dp[i][k]+dp[k+1][j]+a[k+1].x-a[i].x+a[k].y-a[j].y;
                    if (dp[i][j]>now){
                        dp[i][j]=now;
                        pos[i][j]=k;
                    }
                }
            }
        }
        printf("%d\n",dp[1][n]);
    }
}
```

wqs 二分(带权二分)

```
//2018 南京 B，使用的性质是每次更新的 value 斜率会下降
//所以二分这个斜率即可；这个题套了个斜率优化
struct node {
    ll ans; int c;
} dp[maxn];
struct range {
    int l,p;
} Q[maxn];
ll sum[maxn];
int A[maxn];
inline ll calc(int l,int r) {
    ll ret=dp[l].ans;
    l++; int mid=(l+r+1)/2;
    ret+=(sum[r]-sum[mid]-(ll)(r+l-mid*2)*A[mid])-
        (sum[mid-1]-sum[l-1]);

    // printf("%d-%d: ret=%lld  = %lld - %lld\n",l,r,ret,
    //          (sum[r]-sum[mid]-(r-mid)*A[mid]),
    //          (sum[mid-1]-sum[l-1]-(mid-l)*A[mid]));
    return ret;
}
inline bool check(int a,int b,int p) {
    return calc(a,p)>calc(b,p);
}
inline int solve_slope(ll x) {
    int st=0,ed=-1,i;
    Q[++ed]=range{1,0}; Q[ed+1].l=n+1;
    FOR(i,1,n) {
        if (st<=ed&&Q[st+1].l-1<i) st++;
        dp[i]= {calc(Q[st].p,i)+x,dp[Q[st].p].c+1};
        //
printf("%d->%d : %lld\n",Q[st].p,i,dp[i].ans-dp[Q[st].p].ans);
        if (i==n) return dp[n].c;
        if (!check(i,Q[ed].p,n)) {
            while (st<=ed&&!check(i,Q[ed].p,Q[ed].l)) ed--;
            if (st>ed) {
                Q[++ed]= {i+1,i}; Q[ed+1].l=n+1;
            } else {
                // printf("CHECK;\n");
                int l=Q[ed].l,r=n+1;
                while (l+1<r) {
                    int mid=l+(r-l)/2;
                    if (check(i,Q[ed].p,mid)) l=mid;
                    else r=mid;
                } Q[++ed]=range{r,i};
                Q[ed+1].l=n+1;
            }
        }
        // int k; FOR(i,st,ed)
        // printf(" x=%lld; %d - %d : %d;
dp=%lld-%d\n",x,Q[ed].l,n,Q[ed].p,dp[i].ans,dp[i].c);
    }
    return 0;
}
inline ll solve(int m) {
    ll l=-100000000,r=1000000001000000000;
    while (l+1<r) {
        ll mid=l+(r-l)/2;
        if (solve_slope(mid)<m) r=mid;
        else l=mid;
    } solve_slope(l);
    // printf(" ans=%lld %d\n",dp[n].ans,dp[n].c);
    return dp[n].ans-l*m;
}
int main() {
    int i,k;
    scanf("%d%d",&n,&k);
    FOR(i,1,n) scanf("%d",&A[i]),sum[i]=A[i]+sum[i-1];
    // solve_slope(100);
    printf("%lld\n",solve(k));
}
```

数位 DP

```
//当板子了
//这道题是连续的差最大是 1
//需要注意时间空间限制,有时需要 hash
//注意取模时底下 calc 也要取-_-
LL f[27][17][2];
int value[27];
LL calc(int x,int prev,int not_0,int flag) {
    if (x==0) return 1;
    if (!flag&&f[x][prev][not_0]!=-1)
        return f[x][prev][not_0];
    LL ret=0; int i,maxi=9;
    if (flag) maxi=min(maxi,value[x]);
    FOR(i,0,maxi) {
//          if (not_0||i)//这是与 lead_0 有关的写法
        if (not_0&&abs(prev-i)<2) continue;
        else ret+=calc(x-1,i,not_0||i,flag&&(i==maxi));
    } if (!flag) f[x][prev][not_0]=ret;
    return ret;
} LL calc(LL x) {
    int length=0;
    while (x) value[++length]=x%10,x/=10;
    return calc(length,0,0,1);
} LL calc(LL l,LL r) {
    return calc(r)-calc(l-1);
}
int n,m,i,j,T;
int main() {
    memset(f,0xff,sizeof(f));
    FOR(i,1,10000)
        if (calc(i,i)) printf("%d ",i);
    puts("");
    LL l,r;
    scanf("%lld%lld",&l,&r);
    printf("%lld\n",calc(l,r));
}
```

树形依赖背包

```
// 树形依赖背包
// 题意: 是否存在块的 val=i
// 做法: 先树分治变成必须包含 top
// 然后往下 dp, 按照 dfs 序看, 有一段是不能用的
// 所以倒着来 dp 或, 从下往上算贡献
// 大概做法是考虑这个点必选, 所以整体往右移 val[x]来 dp
int A[maxn];
vector<int> edge[maxn];
int sz[maxn];
bool mark[maxn];
int minweight,root;
void dfs1(int x,int fa,int n) {
    int weight=0;
    sz[x]=1;
    for (int v:edge[x]) {
        if (v==fa||mark[v]) continue;
        dfs1(v,x,n); sz[x]+=sz[v];
        weight=max(weight,sz[v]);
    } weight=max(weight,n-sz[x]);
    if (weight<minweight) root=x,minweight=weight;
}
bitset<100007> now[3007],ans;//depth
void dfs2(int x,int fa,int dep) {
    now[dep]=now[dep-1]; sz[x]=1;
    for (int v:edge[x]) {
        if (v==fa||mark[v]) continue;
        dfs2(v,x,dep+1); sz[x]+=sz[v];
    } now[dep-1]|=now[dep]<<A[x];
}
void dfs3(int x) {
    debug("dfs3:%d\n",x);
    now[0].reset(); now[0].set(0);
    dfs2(x,0,1); mark[x]=1;
    ans|=now[0];
    for (int v:edge[x]) {
        if (mark[v]) continue;
        minweight=sz[v];
        dfs1(v,0,sz[v]);
        dfs3(root);
    }
}
int main() {
	int n,m,T;
	int i;
	scanf("%d",&T);
	while (T--) {
		scanf("%d%d",&n,&m);
		REP(i,n-1) {
			int u,v;
			scanf("%d%d",&u,&v);
			edge[u].push_back(v);
			edge[v].push_back(u);
		} FOR(i,1,n) scanf("%d",&A[i]);
		minweight=n;
		dfs1(1,0,n); dfs3(root);
		FOR(i,1,m) printf("%d",(int)ans[i]);
		puts("");
		ans.reset();
		FOR(i,1,n) edge[i].clear(),mark[i]=0;
	}
	return 0;
}
```

DP 套 DP

```
//题意:麻将胡牌的可能种数
//为了不数漏,方法是这样的:
//首先考虑每个可能情况选择的个数,只可能有 3*3*2=18 种
//然后我们把状态压一下,每种牌型可能的 1<<18 的状态!
//对这个 1<<18 的状态进行转移
void print2(int x) {
	int i;
	rREP(i,18) putchar(((x>>i)&1)+'0');
} int encode(int n_2,int n_1,int have2) { //start from n-2 | n-1
	int ret=0;
	ret=ret*3+n_2;
	ret=ret*3+n_1;
	ret=ret*2+have2;
	return ret;
} void decode(int e,int &n_2,int &n_1,int &have2) {
	have2=e%2; e/=2;
	n_1=e%3; e/=3;
	n_2=e%3; e/=3;
}
void printstatus(int e) {
	int n_2,n_1,have2;
	decode(e,n_2,n_1,have2);
	printf(" %d %d %d ",n_2,n_1,have2);
}
int getnextstatus(int status,int k) {
	int nxtstatus=0,n;
	int n_2,n_1,have2;
	int x_2,x_1,xave2;
	REP(n,18) if ((status>>n)&1) {
		decode(n,n_2,n_1,have2);
		x_2=n_1; x_1=k-n_2-n_1; xave2=have2;
		if (x_1>=0) {
			int x=encode(x_2,x_1%3,xave2);
			nxtstatus|=(1<<x);
//
printstatus(n);printf("->");printstatus(x);printf("(+%d)",k);puts("")
;
		} if (!have2&&x_1-2>=0) {
			int x=encode(x_2,x_1-2,1);
			nxtstatus|=(1<<x);
//
printstatus(n);printf("->");printstatus(x);printf("(+%d)",k);puts("")
;
		}
	}
//	printf("get:%d->%d (k=%d)\n",status,nxtstatus,k);
	return nxtstatus;
}
queue<int> Q;
int id[1<<18|7],val[1007];
int tot;
int nxt[1007][7];
void initDP() {
	int i,j; tot=0;
	int k;//this_number
	Q.push(1); id[0]=++tot;
	while (Q.size()) {
		int status=Q.front(); Q.pop();
		FOR(k,0,4) { //只考虑这里产生 2~
			int nxtstatus=getnextstatus(status,k);
			if (!id[nxtstatus])
id[nxtstatus]=++tot,val[tot]=nxtstatus,Q.push(nxtstatus);
```

```
                nxt[id[status]][k]=id[nxtstatus];
            }
        }
//      printf("%d\n",tot);
//      REP(i,(1<<18)) if (id[i]){
//          printf("(%-2d): ",id[i]);
//          print2(i);puts("");
//          REP(j,18) if ((i>>j)&1) printstatus(j);puts("");
//      }
//      FOR(i,1,tot){
//          printf(" %-2d : ",i);
//          print2(val[i]);puts("");
//          REP(j,18) if ((val[i]>>j)&1) printstatus(j);puts("");
//      }
}
int dp[207][207][78];
inline void update(int &x,int y) {
    ((x+=y)>M)&&(x-=M);
}
int solve(int n,int m) {
    int i,j,k,t;
    FOR(i,0,n+3) FOR(j,0,m) FOR(t,0,68) dp[i][j][t]=0;
    dp[0][0][1<<id[encode(0,0,0)]]=1;
    FOR(i,0,n+3) {
        int MAX;
        if (i<n) MAX=4; else MAX=0;
        FOR(j,0,m) {
            FOR(t,1,tot) if (dp[i][j][t]) {
                FOR(k,0,MAX) {
                    int nxtpos=nxt[t][k];
//                      printf("%d->%d;
k=%d\n",t,id[nxtstauts],k);
                    update(dp[i+1][j+k][nxtpos],dp[i][j][t]);
                }
            }
        }
    } int ret=0;
//  FOR(t,1,tot) printf("%d: %d\n",t,dp[n+3][m][t]);
    FOR(t,1,tot) {
        if ((val[t]>>encode(0,0,1))&1) {
            update(ret,dp[n+3][m][t]);
//              printf("t=%d\n",t);
        }
    }
    return ret;
}
int main() {
    int T;
    initDP();
    scanf("%d",&T);
    while (T--) {
        int n,m;
        static int x=0;
        scanf("%d%d",&n,&m);
        printf("Case #%d: %d\n",++x,solve(n,m));
    }
    return 0;
}
```

插头 DP

主要分两种 dp 方式

最小表示法:

```
//插头 dp 长这样
//      ____
//___|o|->
//L==U 就是环个数
template<typename T1,typename T2> struct hashmap {
    const static int seed=199991;
    const static int maxn=1e6+7;
    struct node {
        T1 key; T2 val; int next;
        node() {};
        node(T1 k,T2 v,int n):key(k),val(v),next(n) {};
    } T[maxn]; //更好地空间局部性?(雾)
    int head[seed],size;
    void clear() {
        memset(head,-1,sizeof(head));
        size=0;
    }
    void insert(T1 pos,T2 val) {
        int x=pos%seed;
        T[size]=node(pos,val,head[x]);
        head[x]=size++;
    }
    bool find(int x) {
        for (int i=head[x%seed]; ~i; i=T[i].next)
            if (T[i].key==x) return 1;
        return 0;
        }
        T2 &operator [](T1 x) {
            for (int i=head[x%seed]; ~i; i=T[i].next)
                if (T[i].key==x) return T[i].val;
            insert(x,INF);
            return T[size-1].val;
        }
    };
    typedef hashmap<int,int> HASHMAP;
    HASHMAP MP[2];
    inline int getBit(int x,int k) {
        return (x>>(k+k))&3;
    }
    inline int setBit(int x,int k,int v) {
        return (x&~(3<<(k+k)))|(v<<(k+k));
    }
    inline void insert(HASHMAP &nxt,int k,int val) {
        int &nxtval=nxt[k];
        nxtval=min(nxtval,val);//down
    }
    inline void insert(HASHMAP &nxt,int k,int j,int down,int right,int
val) {
        k=setBit(k,j-1,down);
        k=setBit(k,j,right);
        int &nxtval=nxt[k];
        nxtval=min(nxtval,val);//down
    }
    //题意：要从上往下划个线,把 L 和 R 分开；不能有环
    char str[27][27];
    int main() {
        int n,m;
        int i,j;
        while (~scanf("%d%d",&n,&m)) {
            FOR(i,1,n) scanf("%s",str[i]+1);
            int now=0,nxt=1;
            MP[now].clear();
            FOR(j,1,m) {
                int x=setBit(0,j,1);
                MP[now].insert(x,0);
            };//top_插头
            FOR(i,1,n) {
                FOR(j,1,m) {//check_position; to_right
                    MP[nxt].clear();
                    int more=str[i][j]-'0';
                    for (int it=0; it<MP[now].size; it++) { //x_left;
y:down
                        int
k=MP[now].T[it].key,val=MP[now].T[it].val;
                        int L=getBit(k,j-1),U=getBit(k,j);//v=value
                        int z=0;//from left; downval_count
                        {int t; REP(t,j) if (getBit(k,t)) z^=1;}
                        if
((str[i][j]=='#'||str[i][j]=='W'||str[i][j]=='L')&&(L||U)) {
                            continue;//有插头
                        } else if (str[i][j]=='W') {
                            if (z) continue;
                            insert(MP[nxt],k,MP[now][k]);//no way
                        } else if (str[i][j]=='L') {
                            if (!z) continue;//no!
                            insert(MP[nxt],k,MP[now][k]);//no way
                        } else if (str[i][j]=='#') {//all is ok
                            insert(MP[nxt],k,MP[now][k]);//no way
                        } else {
                            if (L&&U) {//value:(left and up)
                                if (L==U) continue;
                                {
                                    //merge_多个
                                    int t,_k=0,c=0;
                                    int v[10],id[4]={0,0,0,0};
                                    REP(t,m+1) {
                                        v[t]=getBit(k,t);
                                        if (v[t]==L) v[t]=U;
                                    } v[j-1]=v[j]=0;
                                    REP(t,m+1) if (v[t]){
                                        if (!id[v[t]]) id[v[t]]=++c;
                                        _k=setBit(_k,t,id[v[t]]);
                                    } insert(MP[nxt],_k,val+more);
                                }
                            } else if (L||U) {//left or up
                                insert(MP[nxt],k,j,L|U,0,val+more);
                                insert(MP[nxt],k,j,0,L|U,val+more);
                            } else {//circle
                                insert(MP[nxt],k,val);//not_choose
                                {
                                    //get_value
                                    int t,_k=0,c=0;
                                    int v[10],id[4]={0,0,0,0};
                                    REP(t,m+1) {
                                        v[t]=getBit(k,t);
                                        if (v[t]==L) v[t]=U;
                                    } v[j-1]=v[j]=3;//insert_new
                                    REP(t,m+1) if (v[t]){
                                        if (!id[v[t]]) id[v[t]]=++c;
                                        _k=setBit(_k,t,id[v[t]]);
                                    }
insert(MP[nxt],_k,val+more);//not choose
                                }
                            }
                        }
                    } now^=1; nxt^=1;
                }
                MP[nxt].clear();
                for (int it=0; it<MP[now].size; it++) {
                    int k=MP[now].T[it].key; int w=MP[now].T[it].val;
                    if (!getBit(k,m)) insert(MP[nxt],k<<2,w);
                } now^=1; nxt^=1;
            }
            int ans=INF;
            FOR(j,1,m) {
                int x=setBit(0,j,1);
                ans=min(ans,MP[now][x]);
            }
            if (ans<INF) printf("%d\n",ans);
            else puts("-1");
        }
    }
    /*
    6 8
    88W888L8
    888#W888
    888888L8
    8W8LL#88
    8888888L
    00000W88
```

括号序列:

```
    //插头 dp 长这样
    //      _____
    //___|o|->
    template<typename T1,typename T2> struct hashmap {
        const static int seed=199991;//seed 最好设置小点=_=! 要 clear
        const static int maxn=1e6+7;
        struct node {
            T1 key; T2 val; int next;
            node() {};
            node(T1 k,T2 v,int n):key(k),val(v),next(n) {};
        } T[maxn]; //更好地空间局部性?(雾)
        int head[seed],size;
        void clear() {
            memset(head,-1,sizeof(head));
            size=0;
        }
        void insert(T1 pos,T2 val) {
            int x=pos%seed;
            T[size]=node(pos,val,head[x]);
            head[x]=size++;
        }
        bool find(int x) {
            for (int i=head[x%seed]; ~i; i=T[i].next)
                if (T[i].key==x) return 1;
            return 0;
        }
        T2 &operator [](T1 x) {
            for (int i=head[x%seed]; ~i; i=T[i].next)
                if (T[i].key==x) return T[i].val;
            insert(x,INF);
            return T[size-1].val;
        }
    };
    typedef hashmap<int,int> HASHMAP;
    HASHMAP MP[2];
    inline int getBit(int x,int k) {
        return (x>>(k+k))&3;
    }
    inline int setBit(int x,int k,int v) {//注意这里是返回=_=
        return (x&~(3<<(k+k)))|(v<<(k+k));
    }
    inline void insert(HASHMAP &nxt,int k,int val) {
        int &nxtval=nxt[k];
        nxtval=min(nxtval,val);//down
    }
    inline void insert(HASHMAP &nxt,int k,int j,int down,int right,int
val) {
        k=setBit(k,j-1,down);
        k=setBit(k,j,right);
        int &nxtval=nxt[k];
        nxtval=min(nxtval,val);//down
    }
    //题意: 要从上往下划个线,把 L 和 R 分开; 不能有环
    char str[27][27];
```

```
int n,m;
void printstatus(int k,int i,int j,const char str[]=""){
    printf("%s: %d %d; status=",str,i,j);
    REP(i,m+1) printf("%d",getBit(k,i));
    // system("pause");
}
int main() {
    int i,j;
    while (~scanf("%d%d",&n,&m)) {
        FOR(i,1,n) scanf("%s",str[i]+1);
        int now=0,nxt=1;
        MP[now].clear();
        FOR(j,1,m) {
            int x=setBit(0,j,3);
            MP[now].insert(x,0);
        };//top_插头
        FOR(i,1,n) {
            FOR(j,1,m) {//check_position; to_right
                MP[nxt].clear();
                int more=str[i][j]-'O';
                for (int it=0; it<MP[now].size; it++) { //x_left; y:down
                    int k=MP[now].T[it].key,val=MP[now].T[it].val;
                    int L=getBit(k,j-1),U=getBit(k,j);//v=value
                    int z=0;//from left; downval_count
                    {int t; REP(t,j) if (getBit(k,t)) z^=1;}
                    if ((str[i][j]=='#'||str[i][j]=='W'||str[i][j]=='L')&&(L||U)) {
                        continue;//有插头
                    } else if (str[i][j]=='W') {
                        if (z) continue;
                        insert(MP[nxt],k,MP[now][k]);//no way
                    } else if (str[i][j]=='L') {
                        if (!z) continue;//no!
                        insert(MP[nxt],k,MP[now][k]);//no way
                    } else if (str[i][j]=='#') {//all is ok
                        insert(MP[nxt],k,MP[now][k]);//no way
                    } else {
                        if (L&&U) {//value:(left and up)
                            if (L!=U) {
                                if (L==2&&U==1) {// (value= _|)
                                    insert(MP[nxt],k,j,0,0,val+more);
                                } else if (L==3&&U==1) {
                                    int pos,_k=setBit(k,j-1,0);
                                    _k=setBit(_k,j,0);
                                    FOR(pos,j+1,m) if (getBit(k,pos)==2) break;
                                    if (0<=pos&&pos<=m) {
                                        _k=setBit(_k,pos,3);
                                        insert(MP[nxt],_k,val+more);
                                    } else printstatus(k,i,j,"bug1");
                                } else if (L==2&&U==3) {
                                    int pos,_k=setBit(k,j-1,0);
                                    _k=setBit(_k,j,0);
                                    rFOR(pos,0,j-2) if (getBit(k,pos)==1) break;
                                    if (0<=pos&&pos<=m) {
                                        _k=setBit(_k,pos,3);
                                        insert(MP[nxt],_k,val+more);
                                    } else printstatus(k,i,j,"bug2");
                                } else continue;//L=1&&U=2;
                                //merge_circle((i,j)=bottom_right(ex,ey))
                            } else {
                                int pos,_k=setBit(k,j,0);
                                _k=setBit(_k,j-1,0);
                                if (L==1) {
                                    FOR(pos,j+1,m) if (getBit(k,pos)==(L^3)) break;
                                } else {
                                    rFOR(pos,0,j-2) if (getBit(k,pos)==(L^3)) break;
                                } if (0<=pos&&pos<=m) {
                                    _k=setBit(_k,pos,L);
                                    insert(MP[nxt],_k,val+more);
                                } else printstatus(k,i,j,"bug3");
                            }
                        } else if (L||U) {//left or up
                            insert(MP[nxt],k,j,L|U,0,val+more);
                            insert(MP[nxt],k,j,0,L|U,val+more);
                        } else {
                            insert(MP[nxt],k,val);//not_choose
                            insert(MP[nxt],k,j,1,2,val+more);//new
                        }
                    }
                } now^=1; nxt^=1;
            }
            MP[nxt].clear(); //to_next(->)
            for (int it=0; it<MP[now].size; it++) {
                int k=MP[now].T[it].key; int w=MP[now].T[it].val;
                if (!getBit(k,m)) insert(MP[nxt],k<<2,w);
            } now^=1; nxt^=1;
        }
        int ans=INF;
        FOR(j,1,m) {
            int x=setBit(0,j,3);
            ans=min(ans,MP[now][x]);
        }
        if (ans<INF) printf("%d\n",ans);
        else puts("-1");
    }
}
```

斯坦纳树，子集卷积的计数 DP

斯坦纳树:

//题意：有几个点必须连接

//每个边的长度是 1，问你斯坦纳树有几个

// 斯坦纳树，求 min_length 很简单.. min_cnt 会重复计算，所以从小到大计算

// len=1，求方案数

```
struct info {
    int min,cnt;
    info(int _min=INF,int _cnt=0):min(_min),cnt(_cnt) {};
} f[1<<12|7][57],g[1<<12|7][57];
inline void add(info &A,info B) {
    if (A.min>B.min) A=info(B.min,0);
    if (A.min==B.min) add_(A.cnt,B.cnt);
}
inline info merge(info A,info B) {
    info ret(A.min+B.min,(ll)A.cnt*B.cnt%M);
    if (ret.min>n) ret.min=n,ret.cnt=0;
    return ret;
}
vector<int> edge[maxn];
vector<int> have[maxn];
int now[maxn],dep[maxn],vis[maxn];
int TaskA() {
    int i,j,_,maxs;
    scanf("%d",&_); maxs=1<<_;
    REP(i,n) edge[i].clear();
    REP(i,m) {
        int u,v;
        scanf("%d%d",&u,&v);
        u--; v--;
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    REP(i,maxs) REP(j,n) f[i][j]=g[i][j]=info(n,0);
    REP(i,n) {
        int cur=i<_?1<<i:0; vis[i]=-1;
        f[cur][i]=g[cur][i]=info(0,1);
    }
    int sta;
    REP(sta,maxs) {
        REP(i,n) {//f:last_op:addedge; g:no_limit
            if (i<_&&!((sta>>i)&1)) continue;
            int remove=i<_?1<<i:0; int remain=sta^remove;
            int lowbit=remain&-remain; // 防止重复计算，一定注意这里是 remain!
            if (remain)
                for (int pre=remain&(remain-1); pre; pre=remain&(pre-1)) if (pre&lowbit)
add(g[sta][i],merge(f[pre|remove][i],g[(sta^pre)|remove][i]));
            dep[i]=g[sta][i].min;
            if (dep[i]<n) have[dep[i]].push_back(i);
        } //?被卡常了?
        vector<int> Q;
        REP(i,n) {
            for (auto x:have[i]) {
                if (vis[x]==sta) continue;
                Q.push_back(x); vis[x]=sta;
            } for (auto x:Q) {
                info now=info(g[sta][x].min+1,g[sta][x].cnt);
                for (auto v:edge[x]) {
                    if (!(v<_&&!((sta>>v)&1))) {
                        if (dep[v]>dep[x]+1) {
                            dep[v]=dep[x]+1;
                            have[dep[v]].push_back(v);
                        }
                    } int nxtsta=v<_?sta|(1<<v):sta;
                    add(g[nxtsta][v],now);
add(f[nxtsta][v],now);
                }
            } Q.clear(); have[i].clear();
        }
    } //
printf("%d %d\n",g[maxs-1][1].min,g[maxs-1][1].cnt);
    printf("%d\n",g[maxs-1][1].cnt);
    return 0;
}
```

另一个题:

//题意:

//给一堆边，每个生成树上的边贡献 w[i]*max(dep[u],dep[v])

//问你生成树总贡献

//做法：枚举生成树，然后直接 dp 两边 cnt 和 len 得到答案

```
//f:\sum{dep} g:\sum{cnt}
int e[17][17]; int ew[17][17];
int f[17][1<<12|7],g[17][1<<12|7];
int F[17][1<<12|7],G[17][1<<12|7];//F,G:link
int bit[1<<12|7];
int main() {
    int i,j;
    scanf("%d%d",&n,&m);
    REP(i,m) {
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        u--; v--; e[u][v]++; e[v][u]++;
        ew[u][v]+=w; ew[v][u]+=w;
    } int sta;
    REP(i,n) g[i][1<<i]=1;
    REP(sta,(1<<n)) bit[sta]=bit[sta>>1]+(sta&1);
    REP(sta,(1<<n)) {
        REP(i,n) if ((sta>>i)&1) { //this_root
            int remain=sta^(1<<i);
            if (remain){
                int low=remain&-remain;//low 写错了 =_=
                for (int now=remain; now ;
now=(now-1)&remain) if (now&low){
                    int sta1=now,sta2=sta^sta1;
                    add_(f[i][sta],(ll)F[i][sta1]*g[i][sta2]%M);
                    add_(f[i][sta],(ll)G[i][sta1]*f[i][sta2]%M);

add_(g[i][sta],(ll)G[i][sta1]*g[i][sta2]%M);
                }
            } else g[i][sta]=1;
            REP(j,n) if (!((sta>>j)&1)&&e[i][j]){

add_(F[j][sta],e[i][j]*(f[i][sta]+(ll)g[i][sta]*bit[sta]%M)%M);
                add_(G[j][sta],(ll)e[i][j]*g[i][sta]%M);
            }
        }
    } sta=(1<<n)-1; int ans=0;
    REP(i,n) REP(j,n) if (ew[i][j]&&i!=j){
        int s=sta^(1<<j);
        for (int now=s; now; now=(now-1)&s) if
((now>>i)&1){
            int sta1=now,sta2=sta^sta1;
            int
cnt=(f[i][sta1]+(ll)bit[sta1]*g[i][sta1]%M)%M*g[j][sta2]%M;
            add_(ans,(ll)ew[i][j]*cnt%M);
        }
    } printf("%d\n",ans);
}
```

字符串

KMP|最小表示法

```
//记得 border 是个等差数列
int fail[maxn];
int check(char a[],int n){
    fail[0]=fail[1]=0;
    int i,j;
    FOR(i,2,n){
        j=fail[i-1];
        while (j&&a[j+1]!=a[i]) j=fail[j];
        if (a[j+1]==a[i]) fail[i]=j+1;
        else fail[i]=0;
    }if (n%(n-fail[n])==0) return n/(n-fail[n]);
    return 1;
}
//最小表示暴力法
int getmin(char a[],int n){//1-start
    int i,j,l;
    FOR(i,1,n) a[i+n]=a[i];
    i=1,j=2;
    while (i<=n&&j<=n){
        REP(l,n) if (a[i+l]!=a[j+l]) break;
        if (l==n) break;
        if (a[i+l]>a[j+l]) swap(i,j);
        j=max(j+l+1,i+1);
    }return i;
}
```

字典树

```
//x xor v->max;
//没注释的是 v<limit
//注释的是 xor 后小于 limit
//计数问题有个套路:
//先算出全部,然后 for 一边容斥
int nxt[maxn*20*10][2],tot;
int cnt[maxn*20*10];
LL xornum,limit;
void Ins(int &now,int k,int val) {
    if (!now) now=++tot;
    cnt[now]+=val;
    if (k==-1) return;
    int c=(xornum>>k)&1;
    Ins(nxt[now][c],k-1,val);
}
LL Que(int now,int k,bool mark) { //mark:have limit
    if (!now||!cnt[now]) return -INFF;
    if (k==-1) return 0;
    int c=(xornum>>k)&1,lim=(limit>>k)&1;
    LL ret=-INFF;
    if (!lim&&mark) {
        return (c<<k)+Que(nxt[now][0],k-1,mark);
//          return Que(nxt[now][c],k-1,mark);
    } else {
ret=(1ll<<k)+Que(nxt[now][c^1],k-1,mark&&!(c&1));
        if (ret<0) ret=Que(nxt[now][c],k-1,mark&&(c&1));
//          ret=(1ll<<k)+Que(nxt[now][c^1],k-1,mark);
//          if (ret<0) ret=Que(nxt[now][c],k-1,0);
    } return ret;
}
```

AC 自动机

```
//HDU2896,匹配多串,查询 id
namespace ACM {
    const int maxn=505*140;
    int next[maxn][98],fail[maxn],len[maxn],tot;
    vector<int> have[maxn];
    void init() {
        tot=0; len[0]=0; fail[0]=0;
        memset(next[0],0,sizeof(next[0]));
    }
    void insert(char s[],int id) {
        int i,n=strlen(s),p=0;
        REP(i,n) {
            int c=s[i]-33;
            if (!next[p][c]) {
                next[p][c]=++tot; len[tot]=len[p]+1;
                have[tot].clear(); fail[tot]=0;
                memset(next[tot],0,sizeof(next[tot]));
            } p=next[p][c];
            } have[p].push_back(id);
        }
        int Q[maxn],ST,ED;
        void buildAC() {
            ST=0; ED=-1; Q[++ED]=0;
            while (ST<=ED) {
                int p=Q[ST++],c;
                REP(c,98) {
                    if (next[p][c]) {
                        fail[next[p][c]]=p?next[fail[p]][c]:0;
                        Q[++ED]=next[p][c];
                    } else next[p][c]=p?next[fail[p]][c]:0;//否则可
能 fail=self
                }
                for (int v:have[fail[p]])
                    have[p].push_back(v);
            }
        }
        void query(char a[],vector<int> &ans) {
            int p=0;
            int n=strlen(a),i;
            REP(i,n) {
                int c=a[i]-33; p=next[p][c];
                for (int v:have[p]) ans.push_back(v);
            }
        }
    }
```

AC 自动机 另一种写法

// 2016 南宁 D

// 复杂度是所有串的 len 和

// 题意：是否存在一个排列，使得能一一对应

// 做法：求每个点前相同 val 的 len 差，然后直接 AC 自动机

// 修改 fail 的写法

```
    namespace ACM {
        const int maxn=1e6+7;
        map<int,int> next[maxn];
        int fail[maxn],len[maxn],tot;
        bool mark[maxn];
        void init() {
            tot=0; len[0]=0; fail[0]=0; mark[0]=0;
next[0].clear();
        }
        void insert(int s[],int n) {
            int i,p=0;
            REP(i,n) {
                int c=s[i];
                if (!next[p].count(c)) {
                    next[p][c]=++tot; len[tot]=len[p]+1;
                    fail[tot]=0; mark[tot]=0;
                    next[tot].clear();
                } p=next[p][c];
            } mark[p]=1;
        }
        int Q[maxn],ST,ED;
        inline int getnext(int x,int c){
            for (;;x=fail[x]){
                if (len[x]+1<=c) c=0;
                if (!x||next[x].count(c)) break;
            } if (next[x].count(c)) return next[x][c];
            return x;
        }
        void buildAC() {
            ST=0; ED=-1; Q[++ED]=0;
            while (ST<=ED) {
                int p=Q[ST++];
                for (auto now:next[p]){
                    int c=now.first,nxt=now.second;
                    if (p) fail[nxt]=getnext(fail[p],c);
                    else fail[nxt]=0;
                    Q[++ED]=nxt;
                } mark[p]|=mark[fail[p]];
            }
        }
        bool query(int a[],int n) {
            int p=0,have=0,i;
            REP(i,n) {
                int c=a[i]; p=getnext(p,c);
                have|=mark[p];
            } return have;
        }
    }
```

后缀数组

```
    int wa[maxn],wb[maxn],wv[maxn],ws1[maxn];
    int cmp(int *r,int a,int b,int l) {
        return r[a]==r[b]&&r[a+l]==r[b+l];
```

```
}
//sa->pos(后缀排名->pos)
void da(int *r,int *sa,int n,int m) {
    r[n++]=0;//使 rank 从 1 开始(sa[0]=n)
    int i,j,p,*x=wa,*y=wb,*t;
    REP(i,m) ws1[i]=0;//pre-cmp
    REP(i,n) ws1[x[i]=r[i]]++;//r->x
    rep(i,1,m) ws1[i]+=ws1[i-1];
    rREP(i,n) sa[--ws1[x[i]]]=i;//sort(计数排序)
    for (j=1,p=1; p<n; j<<=1,m=p) { //j->2^x
        p=0; rep(i,n-j,n) y[p++]=i; //最后 j 个是不用加(显然)
        REP(i,n) if (sa[i]>=j) y[p++]=sa[i]-j;//后缀顺序
        REP(i,n) wv[i]=x[y[i]];//x+y->wv(由于后缀顺序)
        REP(i,m) ws1[i]=0;
        REP(i,n) ws1[wv[i]]++;
        rep(i,1,m) ws1[i]+=ws1[i-1];
        rREP(i,n) sa[--ws1[wv[i]]]=y[i];//sort(计数排序)
        t=x,x=y,y=t;
        p=1; x[sa[0]]=0;
        rep(i,1,n) x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
    }
}

int rnk[maxn],height[maxn];
void calheight(int *r,int *sa,int n) {
    int i,j,k=0;
    FOR(i,1,n) rnk[sa[i]]=i;
    REP(i,n) {
        if (k) k--;
        j=sa[rnk[i]-1];
        while (r[i+k]==r[j+k]) k++;
        height[rnk[i]]=k;
    }
}
```

后缀自动机

// 1 题意:至少在 k 个子串中出现的子串数量

// 2 题意:sigma{循环后匹配 cnt}

// 这里的 len 不可以直接使用~ 原因是这里的 len 指的是原串 len

// fail 过后,len 是可以直接使用的~ (会 fail 到确定的节点上)

// 这个 fail 的含义是说后缀相同,向前拓展的 val(一个一个拓展 len 差项)

// sam 反向不为拓扑序!注意自己进行拓扑排序

// 更新时注意 len 的限制!(因为更新时可能根本没有考虑前缀 len)

// 注意 nq 在更新时更新时 val 和 q 是相等的,也就是说,维护值时 nq 要完全和 q 一样

// sum{len[x]-len[fail[x]]}=不同串个数,每个串代表 fail->this 的 len

// 每个串的位置建议存的时候就保留下来~ 要不就有点麻烦了

// 复制出来的虚拟节点在计算次数时不参与计算~

// 也就是说计算相同串个数时,复制出来的只是个虚拟的节点

// query 时在末尾加个 0 可以去掉很多的判断!

// 加空字符时注意 len,这个 len 有两个作用:避免 topo 排错,减少 add 特判

// 加的不是 root,就是个空字符,dfs 的话只能 dfs 一个串!从后往前递推可行

// 如果是在一颗树上建,那么直接计数排序按 len 排是错的!一定注意!

// 注意看子串时的重复~

// 小技巧:由于每个节点对应的 len 是一定的,如果想要找 l->r 对应串可以倍增来找到对应的串

// 用 fail 建后缀树时,压缩路径第一个位置为 pos[i]-len[fail[i]]

// 注意一件事:我这样做是并不能保证 len[fail]!=len 的

// 只有 bfs trie 可以保证,这样来进行按 fail 排序建立后缀树

// dfs trie 的时间复杂度是 trie 叶结点深度和=_=!证明..直接当多个

// 只有 bfs 能稳定的保证复杂度,但是好像没人这样卡人

```
struct SAM{
    int next[maxn][26],fail[maxn],len[maxn];
    int cnt,last;
    void init(){
        cnt=last=0;fail[0]=-1;len[0]=0;
        memset(next[0],0,sizeof(next[0]));
    }
    void add(int c){
        int np=++cnt,p=last;
        memset(next[np],0,sizeof(next[np]));
        len[np]=len[p]+1;
        for (;p!=-1&&!next[p][c];p=fail[p]) next[p][c]=np;
        if (p==-1) fail[np]=0;
        else {
            int q=next[p][c];
            if (len[p]+1==len[q]) fail[np]=q;
            else{
                int nq=++cnt;len[nq]=len[p]+1;
                memcpy(next[nq],next[q],sizeof(next[q]));
                fail[nq]=fail[q];
                fail[np]=fail[q]=nq;
                for (;p!=-1&&next[p][c]==q;p=fail[p])
next[p][c]=nq;
```

```
            }
        }
        last=np;
    }
    // 1:trie 上建树,启发式合并 set
    map<int,int> have[maxn];
    int Next[maxn][26],Last[maxn],tot;
    void add(char a[],int id){
        int n=strlen(a),i,p=0;last=0;
        REP(i,n) {
            int c=a[i]-'a';
            if (Next[p][c]) p=Next[p][c],last=Last[p];
            else add(c),Last[p=Next[p][c]=++tot]=last;
            have[last][id]++;
        }
    }
    void merge(map<int,int> &A,map<int,int> &B){
        if (A.size()<B.size()) swap(A,B);
        for (auto now:B) A[now.first]+=now.second;
        B.clear();//delete &B;
    }
    vector<int> edge[maxn];
    LL Ans[maxn];
    void DFS(int x,int k){
        for (int v:edge[x]){DFS(v,k);merge(have[x],have[v]);}
        if (have[x].size()>=k)
            for (auto v:have[x])

Ans[v.first]+=(LL)v.second*(len[x]-len[fail[x]]);
    }
    void solve(int k){
        int i;
        FOR(i,0,cnt) edge[i].clear();
        FOR(i,1,cnt) edge[fail[i]].push_back(i);
        DFS(0,k);
    }
    // 2:在 query 前进行了 cnt[np]++和沿 fail 增加
    set<int> A;int CNT[maxn];
    LL query(char a[]){
        int i;LL ret=0;
        int n=strlen(a),p=0,l=0;A.clear();
        REP(i,n+n-1){
            int c=a[i%n]-'a';
            if (next[p][c]) l++,p=next[p][c];
            else {
                while (p!=-1&&!next[p][c]) p=fail[p];
                if (p==-1) p=l=0;
                else l=len[p]+1,p=next[p][c];
            }while (len[fail[p]]>=n) p=fail[p],l=len[p];
            if (l>=n){
                if (A.count(p)) continue;
                A.insert(p);
                ret+=CNT[p];
            }
            // if (l>=n) printf("i=%2d ret+id(%2d); l=%2d;
+=%d\n",i,p,l,CNT[p]);
        }return ret;
    }

    void print(){
        int i;
        FOR(i,1,cnt) {
        }
    }
    char a[maxn];
    void dfs(int x=0,int len=0){
        int i;
        for (auto v:have[x])
            printf("%2d(%2d) ",v.first,v.second);
        puts("");
        //
printf("%-3d(fail:%-3d,len=%-2d):%s\n",x,fail[x],this->len[x],a);
        REP(i,26){
            if (next[x][i]){
                a[len]=i+'a';
                dfs(next[x][i],len+1);
                a[len]=0;
            }
        }
    }
}sam;
```

<u>后缀自动机+主席树合并</u>

// 查询某串部分在串 l->r 的最大出现次数及位置
// SAM(这个套路)
// 做法:求出后缀树然后直接找到对应位置 merge
// 这里可以看出, fail 的含义就是说
// 某个位置往前 len 差长度的所有子串
// 然后对后缀树来建树然后对 len 倍增
// 就能求出对应的最短对应点来

```
    // SPOJ COT4
    // 题意:S 串后面接字符生成新字符串
    // T 串由两个字符串(T 串)接起来
    // 问你 Tj 在 Si 出现次数
    // (由于是 Si 所以 dfsfail 树,log 查改)
    // 做法是,S 串建 SAM,然后 dfs 出 rank
    // T 串直接考虑在 S 串中的 rank 范围
    struct SAM{
        const static int maxn=2e5+7;
        int
next[maxn][26],fail[maxn],len[maxn],cnt,pos[maxn],Pos[maxn];
        void init(){
            cnt=0;fail[0]=-1;len[0]=0;pos[0]=0;
            memset(next[0],0,sizeof(next[0]));
        }
        int add(int p,int c,int id){
            int np=++cnt;pos[np]=Pos[np]=id;
            memset(next[np],0,sizeof(next[np]));
            len[np]=len[p]+1;
            for (;p!=-1&&!next[p][c];p=fail[p]) next[p][c]=np;
            if (p==-1) fail[np]=0;
            else {
                int q=next[p][c];
                if (len[p]+1==len[q]) fail[np]=q;
                else{
                    int nq=++cnt;len[nq]=len[p]+1;
                    memcpy(next[nq],next[q],sizeof(next[q]));
                    fail[nq]=fail[q];pos[nq]=pos[q];Pos[nq]=0;
                    fail[np]=fail[q]=nq;
                    for (;p!=-1&&next[p][c]==q;p=fail[p])
next[p][c]=nq;
                }
            }
            return np;
        }
        int failnext[maxn][26];
    };
    struct TRIE{
        SAM sam;
        const static int maxn=1e5+26+7;
        void init(){
            sam.init();
            tot=0;ToT=1;id[ToT]=0;val[0]=-1;//1:空
            memset(next[0],0,sizeof(next[0]));
        }
        //1:trie 上建树
        int id[maxn],ToT;//queries
        int next[maxn][26],last[maxn],tot;//on the trie
        int val[maxn];
        void Add(int i,int c){
            int p=id[i];ToT++;
            if (!next[p][c]) {
                next[p][c]=++tot;val[tot]=c;
                memset(next[tot],0,sizeof(next[tot]));
                fa[tot][0]=p;
            }id[ToT]=next[p][c];
        }
        int Q[maxn],st,ed;
        void buildSAM(){
            st=ed=0;Q[ed++]=0;
            while (st!=ed){
                int p=Q[st++];char c;
                REP(c,26) if (next[p][c]){
                    int nxt=next[p][c];
                    last[nxt]=sam.add(last[p],c,nxt);
                    Q[ed++]=nxt;
                }
            }
        }
        //2:get L-R
        int failtot;
        int rank[maxn],sa[maxn];
        void dfsrank(int x){
            if (sam.Pos[x])
rank[sam.Pos[x]]=++failtot,sa[failtot]=sam.Pos[x];
            char c;
            REP(c,26) if (sam.failnext[x][c])
dfsrank(sam.failnext[x][c]);
        }
        void linkfail(){
            int i;

memset(sam.failnext,0,sizeof(sam.failnext[0])*(sam.cnt+1));
            FOR(i,1,sam.cnt)

sam.failnext[sam.fail[i]][val[prev(sam.pos[i],sam.len[sam.fail[i]])]]
=i;
            dfsrank(0);
        }
        //3:build_fa; ladder 长链剖分
        int
fa[maxn][21],son[maxn],top[maxn],len[maxn],dep[maxn];
        vector<int> ladder[maxn],upper[maxn];
        int upp[maxn];
        void buildfa(){
            int i,j,c;
            dep[0]=0;
            FOR(i,1,tot) rep(j,1,21)
                fa[i][j]=fa[fa[i][j-1]][j-1],dep[i]=dep[fa[i][0]]+1;
            rFOR(i,0,tot){
                int o=0;top[i]=i;
                ladder[i].clear();
                REP(c,26) if (next[i][c]){
                    int p=next[i][c];
                    if (!o||len[o]<len[p]) o=p;
                }if (o) len[i]=len[o]+1;else o=0;
                son[i]=o;top[i]=i;
            }FOR(i,0,tot) if (son[i]) top[son[i]]=top[i];
            rFOR(i,0,tot) ladder[top[i]].push_back(i);
            FOR(i,1,tot) if (top[i]==i){
                int u=i;
                REP(j,len[i]){
                    u=fa[u][0];
                    ladder[i].push_back(u);
                    if (!u) break;
                }
            }upp[0]=-1;
            FOR(i,1,tot) upp[i]=upp[i-1]+(i==(i&-i));
        }
        int prev(int x,int k){;
            if (!k) return x;
            x=fa[x][upp[k]];k-=1<<upp[k];
            k-=dep[x]-dep[top[x]];x=top[x];
            return ladder[x][len[x]+k];
        }
    }trie;
    struct queries{
        int op,i,j;char c;
    }q[maxn];
    //3:get Ans_L-R
    int QAQ;
    struct node{
        int l,r,len,next;
        node(){l=r=len=0;}
        node(int _l,int _r,int _len):l(_l),r(_r),len(_len){};
    }A[maxn],C[27];
    node merge(node A,node B){//反着来的,B 在后
        if (A.len==0) return B;
        if (B.len==0) return A;
        if (B.l>B.r||A.l>A.r) return node(0,-1,A.len+B.len);
        int l,r,L,R;
        l=B.r+1;r=B.l-1;
        for(L=B.l,R=B.r;L<=R;){
            int mid=(L+R)/2;
            if (trie.rank[trie.prev(trie.sa[mid],B.len)]<A.l)
L=mid+1;
            else R=mid-1,l=mid;
        }
        for(L=B.l,R=B.r;L<=R;){
            int mid=(L+R)/2;
            if (trie.rank[trie.prev(trie.sa[mid],B.len)]>A.r)
R=mid-1;
            else L=mid+1,r=mid;
        }
        return node(l,r,A.len+B.len);
    }
    //4:solve
    int F[maxn];
    inline int lowbit(int x){return x&-x;}
    void update(int x,int val){
        for (;x<=trie.failtot;x+=lowbit(x)) F[x]+=val;
    }int query(int x){
        int ret=0;
        for (;x;x-=lowbit(x)) ret+=F[x];
        return ret;
    }int query(int l,int r){
        if (l>r) return 0;
        return query(r)-query(l-1);
    }
    node B[maxn];
    int Ans[maxn],head[maxn];
    void addnode(int x,int pos,int i){
        x=trie.id[x];B[i]=A[pos];
        B[i].next=head[x];head[x]=i;
    }
    void getans(int x){
        int i;
        if (x) update(trie.rank[x],1);
        for (i=head[x];~i;i=B[i].next){
            if (B[i].len&&B[i].l<=B[i].r) Ans[i]=query(B[i].l,B[i].r);
        else Ans[i]=0;
    }
    REP(i,26) if (trie.next[x][i]) getans(trie.next[x][i]);
    if (x) update(trie.rank[x],-1);
}
int n,m,Q;
int i,j,k;
char c;
int main(){
    scanf("%d",&Q);
    trie.init();
    FOR(i,1,Q){
        scanf("%d",&q[i].op);
        if (q[i].op==1){
            scanf("%d %c",&q[i].i,&c);q[i].c=c-'a';
            trie.Add(q[i].i,q[i].c);
        }else if (q[i].op==2){
            scanf("%d%d %c",&q[i].i,&q[i].j,&c);q[i].c=c-'a';
        }else scanf("%d%d",&q[i].i,&q[i].j);
    }
    trie.buildfa();
    trie.buildSAM();
    trie.linkfail();
    REP(i,26){
        int l,r,L,R;
        l=trie.failtot+1;r=0;
        for(L=1,R=trie.failtot;L<=R;){
            int mid=(L+R)/2;
            if (trie.val[trie.sa[mid]]<i) L=mid+1;
            else R=mid-1,l=mid;
        }
        for(L=1,R=trie.failtot;L<=R;){
            int mid=(L+R)/2;
            if (trie.val[trie.sa[mid]]>i) R=mid-1;
            else L=mid+1,r=mid;
        }
        C[i]=node(l,r,1);
    }
    FOR(i,0,trie.tot) head[i]=-1;QAQ=1;
    FOR(i,1,Q){
        if (q[i].op==1){
            head[i]=-1;
        }else if (q[i].op==2){
            if (q[i].i==0) A[++QAQ]=merge(C[q[i].c],A[q[i].j]);
            if (q[i].i==1) A[++QAQ]=merge(A[q[i].j],C[q[i].c]);
        }else if (q[i].op==3)
            A[++QAQ]=merge(A[q[i].i],A[q[i].j]);
        else addnode(q[i].j,q[i].i,i);
    }
    getans(0);
    FOR(i,1,Q) if (q[i].op==4) printf("%d\n",Ans[i]);
    return 0;
}
```

马拉车

//p 是每个点为中心的延伸最长回文子串长度，-1 就是原串以这个点为中心的长度

//看到题先去想这种方法，再说其他方法

//区间的最长子串做法是分成两段，然后直接考虑线段树分开算

```
int n,m;
char s[maxn],str[maxn];
int len1,len2,p[maxn],ans;
void init() {
    ans=0; int i;
    str[0]='+'; str[1]='%';
    REP(i,len1+1) {
        str[i*2+2]=s[i];
        str[i*2+3]='%';
    } len2=len1*2+2;
}
// 主要是说已经对称匹配过的不用再进行
void manacher() {
    int id=0,mx=0; int i;
    FOR(i,1,len2-1) {
        if (mx>i) p[i]=min(p[2*id-i],mx-i);
        else p[i]=1;
        while (str[i+p[i]]==str[i-p[i]]) p[i]++;
        if (p[i]+i>mx) {
            mx=p[i]+i; id=i;
        }
    }
}
```

回文自动机

//next 是将字符拼接到两端产生的字符串!

//一定注意这一点!

//也就是说,如果从上到下累积的话,可以很容易的将其与位置联系到一

起!

```
    //注意 last 是可以在线的,但是如果加了个其他的可以从 fail 上爬的,
    //在讨论外边也要向上爬,或者一次过后就保存下来下次接着使用

    //对于 sans,diff,slink:
    //sans 是把之前的 series_ans 保留下来
    //diff 相同时,sans 一定会与上一个相同(由于对称的特殊性)
    //所以只需改变 diff 改变时的 ans 即可

    //区间本质不同回文串数
    //由于 border 的特性, 可以通过等差数列的方法来分类更新答案
    //bzoj5384,跳 border+bit 可以做到两个 log
    struct PAM {
        int next[maxn][27];
        int fail[maxn];
        int len[maxn];//长度
        int diff[maxn];//length(this-fail)
        int anc[maxn];//diff 不同的 fail,共 log 个
        int S[maxn];//字符
        int last;//上一个字符节点
        int n,tot;//n 表示字符位置
        int newnode(int l) {
            memset(next[tot],0,sizeof(next[tot]));
            len[tot]=l;//不是 1...
            return tot++;
        }
        void init() {
            tot=0; last=n=0;
            newnode(0); newnode(-1);
            S[n]=-1; fail[0]=1;
        }
        int getfail(int x) {
            while (S[n-len[x]-1]!=S[n]) x=fail[x];
            return x;
        }
        int add(int c) {
            S[++n]=c;
            int cur=getfail(last);
            if (!next[cur][c]) {
                int now=newnode(len[cur]+2);
                fail[now]=next[getfail(fail[cur])][c];
                diff[now]=len[now]-len[fail[now]];
                if (diff[now]==diff[fail[now]])
                    anc[now]=anc[fail[now]];
                else anc[now]=now;
                next[cur][c]=now;//这里一定要在 fail 后边=_=
            } return last=next[cur][c];
        }
        char a[maxn];
        void dfs(int p,int len=0) {
            int c;
            printf("%-20s (p=%-5d, length=%-5d fail=%-5d
anc=%-5d diff=%-5d)",a,p,this->len[p],fail[p],anc[p],diff[p]);
            // if (p>=2) printf("%d len=%lld\n",);
            puts("");
            REP(c,26) if (next[p][c]) {
                a[len]=c+'a';
                dfs(next[p][c],len+1);
                a[len]=0;
            }
        }
    } pam;
    int dfn[maxn],out[maxn],tot;
    vector<int> edge[maxn];
    void getdfn(int x) {
        dfn[x]=++tot;
        for (int _=0; _<(int)edge[x].size(); _++)
            getdfn(edge[x][_]);
        out[x]=tot;
    }
    namespace SEG {
        int MAX[maxn<<2];
        void init(int val) {
            memset(MAX,0,(val+1)*sizeof(int)*4);
        }
        int query(int x,int l,int r,int L,int R) {
            if (l<=L&&R<=r) return MAX[x];
            int mid=(L+R)/2,ret=0;
            if (l<=mid) ret=max(ret,query(x<<1,l,r,L,mid));
            if (mid<r) ret=max(ret,query(x<<1|1,l,r,mid+1,R));
            return ret;
        }
        void update(int x,int pos,int val,int L,int R) {
            if (L==R) {MAX[x]=val; return;}
            int mid=(L+R)/2;
            if (pos<=mid) update(x<<1,pos,val,L,mid);
            else update(x<<1|1,pos,val,mid+1,R);
            MAX[x]=max(MAX[x<<1],MAX[x<<1|1]);
        }
    }
    namespace BIT {
        int sum[maxn],n;
        void init(int val) {
            memset(sum,0,(val+1)*sizeof(int)); n=val;
        }
        inline int lowbit(int x) {return x&-x;}
        void add(int x,int val) {
            for (; x<=n; x+=lowbit(x)) sum[x]+=val;
        }
        inline int get(int x) {
            int ret=0;
            for (; x; x-=lowbit(x)) ret+=sum[x];
            return ret;
        }
    }
    vector<pair<int,int> > queries[maxn];
    char str[maxn];
    int id[maxn];
    ll ans[maxn];
    int main() {
        int n,q;
        scanf("%d%d%s",&n,&q,str);
        int i;
        pam.init();
        REP(i,n) id[i+1]=pam.add(str[i]-'a');
        // pam.dfs(0); puts("0");//2
        // pam.dfs(1); puts("1");//1
        REP(i,pam.tot) edge[i].clear();
        REP(i,pam.tot) if (i!=1) edge[pam.fail[i]].push_back(i);
        tot=0; getdfn(1);
        FOR(i,1,q) {
            int l,r; scanf("%d%d",&l,&r);
            queries[r].push_back(make_pair(l,i));
        } BIT::init(n);
        SEG::init(tot);
        FOR(i,1,n) {
            // for (int v=T.last;T.len[v]>0;v=T.slink[v]){
            //      sans[v]=f[i-(T.len[T.slink[v]]+T.diff[v])];
            //      if (T.diff[v]==T.diff[T.fail[v]])
            //           (sans[v]+=sans[T.fail[v]])%=M;
            //      if (!(i&1)) (f[i]+=sans[v])%=M;//f[x]
            // }
            for (int p=id[i]; pam.len[p]>0;
p=pam.fail[pam.anc[p]]) {
                int
l=max(1,SEG::query(1,dfn[p],out[p],1,tot)-pam.len[p]+1+1);
                int r=i-pam.len[pam.anc[p]]+1+1;//+1:start; 等
差数列一起算
                BIT::add(l,1); BIT::add(r,-1);
            } SEG::update(1,dfn[id[i]],i,1,tot);
            for (int _=0; _<(int)queries[i].size(); _++)

ans[queries[i][_].second]=BIT::get(queries[i][_].first);
        }
        int Ans=0;
        FOR(i,1,q) Ans=(Ans+(ll)ans[i]*i)%M;
        printf("%d\n",Ans);
    }
```

二分 hash

```
// wannafly 挑战赛 11D
// 题意:求上下拼接后的最长回文串长度(很坑)
struct hashset{
    const static int seed=1e7+7;
    const static int maxn=2e6+7;
    struct node{
        int x,y;int next;
        node(){};
        node(int _x,int _y,int n):x(_x),y(_y),next(n){};
    }T[maxn];//更好地空间局部性?(雾)
    int head[seed],size;
    void clear(){
        memset(head,-1,sizeof(head));
        size=0;
    }
    void insert(int x,int y){
        int& h=head[x%seed];
        for (int i=h;~i;i=T[i].next)
            if (T[i].x==x&&T[i].y==y) return;
        T[size]=node(x,y,h);h=size++;
    }
    bool count(int x,int y){
        for (int i=head[x%seed];~i;i=T[i].next)
            if (T[i].x==x&&T[i].y==y) return 1;
        return 0;
    }
}have;
struct hash{
    int px[maxn],val[maxn],p;
    void setp(int P,int n=200000){
        int i;px[0]=1;p=P;
        FOR(i,1,n) px[i]=(LL)px[i-1]*p%M;
    }
    void set(char a[],int n){
        int i;val[0]=0;
        FOR(i,1,n) val[i]=((LL)val[i-1]*p+a[i-1])%M;
    }
    int get(int l,int r){
        l++;r++;
        int ret=val[r]-(LL)val[l-1]*px[r-l+1]%M;
        (ret<0)&&(ret+=M);return ret;
    }
}HA,RB;
void manacher(char A[],int p[],int len){
    int id=0,mx=0,i;
    rep(i,1,len){
        if (mx>i) p[i]=min(p[2*id-i],mx-i);
        else p[i]=1;
        while (A[i+p[i]]==A[i-p[i]]) p[i]++;
        if (p[i]+i>mx) mx=p[i]+i,id=i;
    }
}
int n,i;
int s[maxn];
char a[maxn],b[maxn],A[maxn*2],B[maxn*2];
int PA[maxn*2],PB[maxn*2];//id
int len,ans;
int main(){
    scanf("%d",&n);
    scanf("%s%s",a,b+1);
    a[n]='(';b[0]=')';n++;
    A[len]='+';B[len]='-';len++;
    A[len]='%';B[len]='%';len++;
    REP(i,n){
        A[len]=a[i];B[len]=b[i];len++;
        A[len]='%'; B[len]='%'; len++;
    }A[len]='*';B[len]='/';len++;
    n=len;
    manacher(A,PA,len);
    manacher(B,PB,len);
    HA.setp(19);RB.setp(19);
    HA.set(A,n);reverse(B,B+n);RB.set(B,n);
    reverse(B,B+n);
    rep(i,1,n){
            //min(i-1-PA[i]+1,n-1-i-PA[i]+1)+1
            //PA 和 PB 的判断相同 (只需一个最大即可)
            PA[i]=max(PA[i],PB[i]);
            int l=0,r=min(i-PA[i],n-1-i-PA[i])+1;//r:not
            while (l+1<r){
                int mid=(l+r)/2;
                int hash_A=HA.get(i-PA[i]-mid+1,i-PA[i]);
                int
hash_B=RB.get(n-(i+PA[i]+mid),n-1-(i+PA[i]));
                if (hash_A==hash_B) l=mid;
                else r=mid;
            }ans=max(ans,PA[i]+l);
        }printf("%d\n",ans-1);
    }
```

一些 hashset|hashmap

```
template<typename T1,typename T2> struct hashmap{
    const static int seed=999991;
    const static int maxn=1e6+7;
    struct node{
        T1 key;T2 val;int next;
        node(){};
        node(T1 k,T2 v,int n):key(k),val(v),next(n){};
    }T[maxn];//更好地空间局部性?(雾)
    int head[seed],size;
    void clear(){
        memset(head,-1,sizeof(head));
        size=0;
    }
    void insert(T1 pos,T2 val){
        int x=pos%seed;
        T[size]=node(pos,val,head[x]);
        head[x]=size++;
    }
    T2 &operator [](T1 x){
        for (int i=head[x%seed];~i;i=T[i].next)
            if (T[i].key==x) return T[i].val;
        insert(x,0);
        return T[size-1].val;
    }
};
```

//用于字典树啥的空间优化

```
struct linknode{
    struct node{
        int key,val;int next;
        node(){};
        node(int k,int v,int n):key(k),val(v),next(n){};
    }T[maxn];//更好地空间局部性?(雾)
    int head[maxn],size;
    void clear(){
        memset(head,-1,sizeof(head));
        size=0;
    }
    int get(int x,int y){
        for (int i=head[x];~i;i=T[i].next)
            if (T[i].key==y) return T[i].val;
        return 0;
    }
    void insert(int pos,int key,int val){
        T[size]=node(key,val,head[pos]);
        head[pos]=size++;
    }
};
```

后缀平衡树

```
// 替罪羊树...这道题卡 splay,treap
// 题意：加字符，减字符，query 子串个数
// 做法：建后缀自动机+LCT; right 集个数
// 后缀自动机做法是直接链加链减
// 或者后缀顺序建平衡树然后树上 query
// 后缀平衡树的顺序是倒着的，倒着的后缀 rank
// 以上是 https://www.nowcoder.net/acm/contest/59/C
// 由于这个是倒着的 rank，反过来的情况非常常见(往前加)
// 这个直接用这个板子 insert, query 即可
const double alpha=0.75;
namespace SAT {
    const ull MAX=(1ull<<63)-1;
    struct node {
        int son[2]; int pre,size;
        int sum,val; ull rank; char c;
        void initval(char _c) {
            son[0]=son[1]=0; pre=0;
            size=sum=val=1; rank=0; c=_c;
        }
    } T[maxn];
    int cnt,root,last;
    inline bool cmp(int x,int y) {//x<y
        assert(x!=y);
        if (T[x].c!=T[y].c) return T[x].c<T[y].c;
        return T[T[x].pre].rank<T[T[y].pre].rank;//same:
    }
    void pushup(int x){
        T[x].size=1; T[x].sum=T[x].val;
        if (T[x].son[0]) {
            T[x].size+=T[T[x].son[0]].size;
            T[x].sum+=T[T[x].son[0]].sum;
        } if (T[x].son[1]) {
            T[x].size+=T[T[x].son[1]].size;
            T[x].sum+=T[T[x].son[1]].sum;
        }
    }
    int id[maxn],tot;
    bool rebuildRoot;//手动 rebuild_{root}
    void getrank(int x) {
        if (T[x].son[0]) getrank(T[x].son[0]);
        if (!rebuildRoot||T[x].val) id[++tot]=x;
        if (T[x].son[1]) getrank(T[x].son[1]);
    }
    void rerank(int &x,int l,int r,ull L,ull R) {
        x=0; if (l>r) return;
        ull mid=(L+R)/2; int m=(l+r)/2;
        x=id[m]; T[x].rank=mid;
        rerank(T[x].son[0],l,m-1,L,mid-1);
        rerank(T[x].son[1],m+1,r,mid+1,R);
        pushup(x);
    }
    void rebuild(int &x,ull l,ull r) {
        if (!x) return;
        tot=0; getrank(x);
        rerank(x,1,tot,l,r);
    }
    void ins(int &x,ull l,ull r) {
        ull mid=(l+r)/2;
        if (!x) {x=cnt; if (l<=r) T[x].rank=mid; return;}
        int p=cmp(x,cnt);
        int &son=T[x].son[p];
        if (p==0) ins(son,l,mid-1);
        else ins(son,mid+1,r);
        pushup(x); //changes
        if (max(T[T[x].son[0]].size,T[T[x].son[1]].size)>
                T[x].size*alpha) rebuild(x,l,r);
    }
void insert(char c) {
    T[++cnt].initval(c);
    T[cnt].pre=last; last=cnt;
    ins(root,1,MAX);
    if (!T[cnt].rank) {
        rebuildRoot=true;
        rebuild(root,1,MAX);
        rebuildRoot=false;
    }
}
void insert(char s[]) {
    int len=strlen(s),i;
    REP(i,len) insert(s[i]);
}
bool cmp(int k,char s[],int len) {//smaller //okay!
    for (int i=0; i<len; i++,k=T[k].pre) {
        if (!k) return 1;
        if (s[i]!=T[k].c) return T[k].c<s[i];
    } return 0;
}
int query(char s[],int len) {
    int ret=0;
    for (int now=root; now;) {
        if (!cmp(now,s,len)) now=T[now].son[0];
        else {
            ret+=T[now].val+T[T[now].son[0]].sum,
                now=T[now].son[1];
        }
    } return ret;
}
int query(char s[]) {
    int len=strlen(s);
    reverse(s,s+len); s[len]='Z'+1;// s[len+1]=0;
    return query(s,len+1)-query(s,len);
}
void del(int k) {
    for (; k&&last; last=T[last].pre,k--) {
        int now;
        for (now=root; now!=last;) {
            T[now].sum--;
            int p=T[last].rank>=T[now].rank;
            now=T[now].son[p];
        } assert(last==now);
        T[last].val=0; T[last].sum--;
    } if (!last) root=0;
    }
    void init(){
        cnt=root=last=0;
    }
}

//2017icpc 青岛 J
//题意：每个串找个后缀拼起来
//query 后缀最小序是多少
//倒着加，然后找个最小 rank 把剩下的都去掉即可
char pool[maxn],*st=pool;
char *A[maxn]; int len[maxn];
char ans[maxn];int L;
int main() {
    int T;
    scanf("%d",&T);
    while (T--){
        int i,j;
        SAT::init(); L=0; st=pool;
        scanf("%d",&n);
        REP(i,n) {
            A[i]=st,scanf("%s",A[i]);
            st+=(len[i]=strlen(A[i]));
        }
        rREP(i,n) {
            // printf("i=%d;\n",i);
            rREP(j,len[i]) SAT::insert(A[i][j]);
            int k=SAT::last; ull MIN=SAT::T[k].rank;int l=0;
            REP(j,len[i]) {//del_cnt
                if (MIN>SAT::T[k].rank)
MIN=SAT::T[k].rank,l=j;
                k=SAT::T[k].pre;
            } SAT::del(l);
            rrep(j,l,len[i]) ans[L++]=A[i][j]; ans[L]=0;
        } reverse(ans,ans+L);
        printf("%s\n",ans);
    }
    return 0;
}
```

子序列自动机

```
//序列自动机：假设 y 之后第一次出现位置为 nxt[x][y]
//f[x]=f[x]+f[nxt[x][y]]; 所以只与 head 是几有关
//trans[head][last(nextvalue)]
```

```
    //保存的是第一个 head 处的值(转移)
    //多加一个节点保存到结尾位置的 ans 是多少
    //题意：给个字符串，每次在原串的每两个字符中间加个字符，问你最
后的子序列个数
    //做法：倒着发现是个复制后中间加个字符，用子序列自动机来做
    int fa[maxn];
    ll val[maxn];
    struct mat{
        int trans[27][27]; bool have[27];
        mat(){memset(trans,0,sizeof(trans));
memset(have,0,sizeof(have));}
    };
    mat mul(mat A,mat B){
        mat ret; int i,j,k;
        REP(i,27) {
            if (A.have[i]) {
                REP(j,27) {
                    if (B.have[j])
                        REP(k,27)
add_(ret.trans[i][k],(ll)A.trans[i][j]*B.trans[j][k]%M);
                    else add_(ret.trans[i][j],A.trans[i][j]);
                }
            } else REP(j,27) add_(ret.trans[i][j],B.trans[i][j]);
        }
        REP(i,27) ret.have[i]=A.have[i]|B.have[i];
        return ret;
    }
    mat getMat(int c){
        c-='a'; mat ret; int i; ret.have[c]=1;
        REP(i,27) ret.trans[c][i]=1;
        return ret;
    }
    int main() {
        int i;
        scanf("%d",&n);
        scanf("%s",str);
        mat ans=getMat(str[n-1]); int Ans=0;
        rREP(i,n-1) {
            // int j,k;;
            // mat mat1=mul(getMat(str[i]),ans);
            // REP(j,27) {
            //   REP(k,27){
            //        printf("%2d ",mat1.trans[j][k]);
            //   } puts("<-mul");
            // }
            ans=mul(ans,mul(getMat(str[i]),ans));
            // REP(j,27) {
            //   REP(k,27){
            //        printf("%2d ",ans.trans[j][k]);
            //   } puts("<-ans");
            // }
        }
        REP(i,26) add_(Ans,ans.trans[i][26]);
        printf("%d\n",Ans);
    }
```

区间 border

```
#include <vector>
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <stack>

#define rep(i, a, n) for (int i = a; i < n; ++i)
#define per(i, a, n) for (int i = n - 1; i >= a; --i)
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef std::vector<int> VI;
typedef long long ll;
typedef std::pair<int, int> PII;

#define gcd(a, b) std::__gcd((a), (b))

const int N = 1000010;
int ff[N], n, a[N], sql[N], sqr[N], sqr2[N];

struct SuffixArray {
    int sa[N], rk[N], ht[N];
    bool t[N << 1];
    int hv[21][N];

    bool islms(const int i, const bool *t) {
        return i > 0 && t[i] && !t[i - 1];
    }

    template<class T>
    inline void sort(T s, int *sa, const int len, const int sz, const int sigma, bool *t,
                     int *b, int *cb, int *p) {
        memset(b, 0, sizeof(int) * sigma);
        memset(sa, -1, sizeof(int) * len);
        rep(i, 0, len) b[(int) s[i]]++;
        cb[0] = b[0];
        rep(i, 1, sigma) cb[i] = cb[i - 1] + b[i];
        per(i, 0, sz) sa[--cb[(int) s[p[i]]]] = p[i];
        rep(i, 1, sigma) cb[i] = cb[i - 1] + b[i - 1];
        rep(i, 0, len) if (sa[i] > 0 && !t[sa[i] - 1]) sa[cb[(int) s[sa[i] - 1]]++] = sa[i] - 1;
        cb[0] = b[0];
        rep(i, 1, sigma) cb[i] = cb[i - 1] + b[i];
        per(i, 0, len) if (sa[i] > 0 && t[sa[i] - 1]) sa[--cb[(int) s[sa[i] - 1]]] = sa[i] - 1;
    }

    template<class T>
    inline void sais(T s, int *sa, const int len, bool *t, int *b, int *b1, const int sigma) {
        int p = -1, *cb = b + sigma;
        t[len - 1] = 1;
        per(i, 0, len - 1) t[i] = s[i] < s[i + 1] || (s[i] == s[i + 1] && t[i + 1]);
        int sz = 0, cnt = 0;
        rep(i, 1, len) if (t[i] && !t[i - 1]) b1[sz++] = i;
        sort(s, sa, len, sz, sigma, t, b, cb, b1);
        sz = 0;
        rep(i, 0, len) if (islms(sa[i], t)) sa[sz++] = sa[i];
        rep(i, sz, len) sa[i] = -1;
        rep(i, 0, sz) {
            int x = sa[i];
            rep(j, 0, len) {
                if (p == -1 || s[x + j] != s[p + j] || t[x + j] != t[p + j]) {
                    ++cnt;
                    p = x;
                    break;
                } else if (j > 0 && (islms(x + j, t) || islms(p + j, t))) {
                    break;
                }
            }
            sa[sz + (x >>= 1)] = cnt - 1;
        }
        for (int i = len - 1, j = len - 1; i >= sz; --i) if (sa[i] >= 0) sa[j--] = sa[i];
        int *s1 = sa + len - sz, *b2 = b1 + sz;
        if (cnt < sz) sais(s1, sa, sz, t + len, b, b1 + sz, cnt);
        else
            rep(i, 0, sz) sa[s1[i]] = i;
        rep(i, 0, sz) b2[i] = b1[sa[i]];
        sort(s, sa, len, sz, sigma, t, b, cb, b2);
    }

    template<class T>
    inline void getHeight(T s, int n) {
        rep(i, 1, n + 1) rk[sa[i]] = i;
        int j = 0, k = 0;
        for (int i = 0; i < n; ht[rk[i++]] = k) {
            for (k ? k-- : 0, j = sa[rk[i] - 1]; s[i + k] == s[j + k]; ++k);
        }
    }

    template<class T>
    inline void init(T s, const int len, const int sigma) {
        sais(s, sa, len, t, rk, ht, sigma);
    }

    inline void solve(int *s, int len) {
        init(s, len + 1, 8);
        getHeight(s, len);
        rk[len] = 0;
        rep(i, 1, len + 1) hv[0][i] = ht[i];
        rep(j, 1, 20) for (int i = 1; i + (1 << j) - 1 <= len; ++i) {
                hv[j][i] = std::min(hv[j - 1][i], hv[j - 1][i + (1 << (j - 1))]);
            }
    }

    int lcp(int p, int q) {
        if (q > n) return 0;
        if (p == q) return n - p + 1;
        p = rk[p - 1];
        q = rk[q - 1];
        if (p > q) std::swap(p, q);
        int w = ff[q - p];
```

```
            return std::min(hv[w][p + 1], hv[w][q - (1 << w) + 1]);
        }
    } s1, s2;

    char s[N];

    void gao(int l, int r, int ty) {
        int pi = l - s2.lcp(n + 1 - r, n + 1 - (l - 1));
        int pj = r + s1.lcp(l, r + 1);
        int p = r - l + 1;
        if (pj - pi + 1 >= 2 * p && pi < l && l <= pi + p && (ty ==
0 || pj != n)) {
            for (int k = pi + 2 * p - 1; k <= pj; ++k) {
                sqr[k] = std::min(sqr[k], 2 * p);
                sql[k - 2 * p + 1] = std::min(sql[k - 2 * p + 1], 2
* p);
                sqr2[k] = std::max(sqr2[k], k - 2 * p + 1);
            }
        }
    }

    void gao(int *a) {
        s1.solve(a + 1, n);
        std::reverse(a + 1, n + a + 1);
        a[n + 1] = 0;
        s2.solve(a + 1, n);
        //std::reverse(a + 1, n + a + 1);
        std::stack<PII> st;
        st.push({n + 1, n + 1});
        per(i, 1, n + 1) {
            int j = i;
            while (SZ(st) > 1) {
                auto seg = st.top();
                if (s1.rk[i - 1] >= s1.rk[seg.fi - 1]) break;
                j = seg.se;
                st.pop();
            }
            st.push({i, j});
            gao(i, j, 0);
        }
        st = std::stack<PII>();
        st.push({n + 1, n + 1});
        per(i, 1, n + 1) {
            int j = i;
            while (SZ(st) > 1) {
                auto seg = st.top();
                if (s1.rk[i - 1] <= s1.rk[seg.fi - 1]) break;
                j = seg.se;
                st.pop();
            }
            st.push({i, j});
            gao(i, j, 1);
        }
    }

namespace border {
    int tmp1[N], tmp2[N], c[N], sa[N], rk[21][N], lev;
    PII pos[21][N];

    void buildDict(char *s, int *sa, int *x, int *y, int n, int m
= 128) {
        rep(i, 0, m) c[i] = 0;
        rep(i, 0, n) c[x[i] = s[i]]++;
        rep(i, 1, m) c[i] += c[i - 1];
        per(i, 0, n) sa[--c[x[i]]] = i;
        rep(i, 0, n) rk[0][i] = x[i];
        rep(i, 0, n) pos[0][i] = {rk[0][sa[i]], sa[i]};
        lev = 1;
        for (int k = 1; k < n; k <<= 1, ++lev) {
            int p = 0;
            per(i, n - k, n) y[p++] = i;
            rep(i, 0, n) if (sa[i] >= k) y[p++] = sa[i] - k;
            rep(i, 0, m) c[i] = 0;
            rep(i, 0, n) c[x[y[i]]] ++;
            rep(i, 1, m) c[i] += c[i - 1];
            per(i, 0, n) sa[--c[x[y[i]]]] = y[i];
            std::swap(x, y);
            p = 1;
            x[sa[0]] = 0;
            y[n] = -1;
            //rep(i, 1, n) if (y[sa[i - 1]] == y[sa[i]]) assert(sa[i
- 1] + k <= n && sa[i] + k <= n);
            rep(i, 1, n) x[sa[i]] = (y[sa[i - 1]] == y[sa[i]] &&
y[sa[i - 1] + k] == y[sa[i] + k]) ? p - 1 : p++;
            rep(i, 0, n) rk[lev][i] = x[i];
            rep(i, 0, n) pos[lev][i] = {rk[lev][sa[i]], sa[i]};
            m = p;
        }
    }
```

```
        struct seq {
            int a, k, sz;
            bool contain(int x) {
                if (sz == 0) return 0;
                if (x < a || x > a + (sz - 1) * k) return 0;
                if (x == a) return 1;
                return (x - a) % k == 0;
            }
        };

        ll Inv(ll q, ll m) {
            if (q == 0) return 0;
            //assert(q >= 0);
            ll a1 = m, b1 = 0, a2 = q, b2 = 1, a3, b3, t;
            while (a2 != 1) {
                t = a1 / a2, a3 = a1 - t * a2, b3 = b1 - t * b2 % m,

                a1 = a2, a2 = a3, b1 = b2, b2 = b3;
                if (b2 < 0) b2 += m;
            }
            return b2;
        }

        std::pair<ll, ll> merge(ll a, ll b, ll c, ll d) {
            c -= a;
            ll dd = gcd(b, d);
            if (c % dd != 0) return {-1, -1};
            b /= dd;
            c /= dd;
            d /= dd;
            ll t = c * Inv(b, d) % d;
            if (t < 0) t += d;
            return {b * t * dd + a, b * d * dd};
        }

        seq intersect(seq a, seq b) {
            if (a.sz > b.sz) std::swap(a, b);
            if (a.sz == 0) return a;
            else if (a.sz == 1) {
                if (b.contain(a.a)) return a;
                else return (seq) {0, 0, 0};
            } else {
                std::pair<ll, ll> d = merge(a.a % a.k, a.k, b.a % b.k, b.k);

                if (d.se == -1) return (seq) {0, 0, 0};
                int l = std::max(a.a, b.a), r = std::min(a.a + (a.sz - 1) * a.k, b.a + (b.sz - 1) * b.k);
                int pl = (l - d.fi + d.se - 1) / d.se, pr = (r - d.fi + d.se) / d.se - 1;
                if (pl > pr) return (seq) {0, 0, 0};
                else return (seq) {(int) (d.fi + pl * d.se), (pl == pr) ? 1 : (int) d.se, pr - pl + 1};
            }
        }

        int findprev(int p, int lev, int r) { // <=r start position
            PII *ps = std::lower_bound(pos[lev], pos[lev] + n, std::make_pair(rk[lev][p], r + 1));
            if (ps != pos[lev]) --ps; else return -1;
            if (ps->fi != rk[lev][p]) return -1;
            else return ps->se;
        }

        int findnxt(int p, int lev, int l) {// >=l
            PII *ps = std::lower_bound(pos[lev], pos[lev] + n, std::make_pair(rk[lev][p], l));
            if (ps == pos[lev] + n || ps->fi != rk[lev][p]) return -1;
            else return ps->se;
        }

        int bit[24];
    //#define bit(k) (1<<(k))
    #define bit(k) bit[k]

        seq occur(int p, int lev, int l, int r) {
            int fp = findnxt(p, lev, l);
            if (fp == -1 || fp > r) return (seq) {0, 0, 0};
            int fq = findnxt(p, lev, fp + 1);
            if (fq == -1 || fq > r) return (seq) {fp, 1, 1};
            int fr = findprev(p, lev, r);
            return (seq) {fp, fq - fp, (fr - fp) / (fq - fp) + 1};
        }

        int query(int l, int r) {
            --l;
            --r;
            for (int k = lev; k >= 1; k--) {
                if ((r - l + 1) <= bit(k - 1)) continue;
                seq a = occur(l, k - 1, std::max(r - bit(k) + 1, l), r - bit(k - 1) + 1);
                seq b = occur(r - bit(k - 1) + 1, k - 1, l, std::min(l
+ bit(k - 1), r - bit(k - 1) + 1));
                a.a = l + r - (a.a + (a.sz - 1) * a.k);
                b.a += bit(k - 1) - 1;
                seq c = intersect(a, b);
                if (c.sz != 0 && c.a + (c.sz - 1) * c.k == r) --c.sz;
                if (c.sz != 0) return c.a + (c.sz - 1) * c.k - l + 1;
            }
            return 0;
        }

        void init() {
            buildDict(s, sa, tmp1, tmp2, n);

            for (int i = 0; i < 24; ++i) bit[i] = 1 << i;
        }
    }

    std::pair<char, char> input[N];
    char tmpa[5], tmpb[5];

    char convert() {
        switch(tmpb[0]) {
            case 'd': return 'a';
            case 'r': return 'b';
            case 'm': return 'c';
            case 'f': return 'd';
            case 's':
                if (tmpb[1] == 'o') return 'e';
                if (tmpb[1] == 'i') return 'g';
            case 'l': return 'f';
        }
        //assert(0);
    }

    void gets() {
        std::deque<char> dq;
        for (int i = 1; i <= n; ++i) {
            scanf("%s%s", tmpa, tmpb);
            input[i] = {tmpa[0], convert()};
            //input[i].first = rand() & 1 ? 'a' : 'p';
            //input[i].second = (rand() % 7) + 'a';
            if (input[i].first == 'a') dq.push_back(input[i].second);
            else dq.push_front(input[i].second);
        }
        for (int i = 0; i < n; ++i) {
            s[i] = dq.front();
            dq.pop_front();
        }
    }

    // 题意:求循环节
    int cntp[N];
    int main() {
        //auto bg = clock();
        rep(i, 2, 1000001) ff[i] = ff[i >> 1] + 1;
        scanf("%d", &n);
        //n = 1000000;
        rep(i, 1, n + 1) sql[i] = n + 1, sqr[i] = n + 1;

        gets();

        rep(i, 1, n + 1) {
            a[i] = s[i - 1] - 'a' + 1;
        }
        //std::cout << clock() - bg << std::endl;
        gao(a);
        rep(i, 1, n + 1) sqr2[i] = std::max(sqr2[i], sqr2[i - 1]);
        border::init();
        //std::cout << clock() - bg << std::endl;

        for (int i = n; i; --i) {
            cntp[i] = cntp[i + 1] + input[i].first == 'p';
        }
        for (int i = 1; i <= n; ++i) {
            int l = cntp[i + 1] + 1;
            int r = l + i - 1;
            int bd = border::query(l, r);
            int tot = r - l + 1;
            int len = tot - bd;
            printf("%d\n", (tot + len - 1) / len);
        }
        //std::cout << clock() - bg << std::endl;
        return 0;
    }
```

数据结构

时间分治

```
// 题意：动态求桥的个数
// 做法：按时间分治，然后缩边，tarjan 找桥
int m;
struct E {
    int v,w;
    E() {}
    E(int _v,int _w) {v=_v; w=_w;}
};
struct node {
    int u,v,w,l,r;
    node(int _u,int _v,int _w,int _l,int _r) {
        u=_u; v=_v; w=_w; l=_l; r=_r;
    }
};
vector<E> edge[maxn];
typedef long long ll;
vector<node> remain;
int key[maxn],dfn[maxn];
E build(int x,int fa) {//build vt
    vector<E> ch; dfn[x]=1;
    for (auto now:edge[x]) {
        if (now.v==fa) continue;
        E w=build(now.v,x); w.w+=now.w;
        if (w.v) ch.push_back(w);
    } if (ch.size()>=2) key[x]=1;
    if (key[x]) {
        for (auto v:ch)
            remain.push_back(node(x,v.v,v.w,1,m));//exist!
        return E(x,0);
    } if (ch.size()) return ch[0];
    return E(0,0);
}
int low[maxn],vis[maxn],id[maxn];
int S[maxn],top,tot;
//first:  more;
void tarjan(int x,int fa) { //先缩这个，再建虚树
    dfn[x]=low[x]=++tot;
    S[++top]=x; vis[x]=1;
    int cntfa=0;
    for (auto now:edge[x]) {
        if (now.v==fa&&!cntfa) {
            cntfa=1; continue;
        }
        if (!dfn[now.v]) {
            tarjan(now.v,x);
            low[x]=min(low[x],low[now.v]);
        } else if (vis[now.v])
            low[x]=min(low[x],dfn[now.v]);
    } if (dfn[x]==low[x]) {
        while (1) {
            int now=S[top--];
            vis[now]=0; id[now]=x;
            if (now==x) break;
        }
    }
}
void clear_and_set(int l,int r,const vector<node> &nodes) {
    for (auto x:nodes) {
        edge[x.u].clear();
        edge[x.v].clear();
        dfn[x.u]=dfn[x.v]=0;
    } tot=0;//clear
    for (auto x:nodes) {
        if (x.l<=l&&r<=x.r) {
            edge[x.u].push_back(E(x.v,x.w));
            edge[x.v].push_back(E(x.u,x.w));
        }
    }//all_have
}
int ans[maxn];
void solve(int l,int r,vector<node> nodes,int base) { //m:边数
    clear_and_set(l,r,nodes);
    for (auto x:nodes) {
        if (!dfn[x.u]) tarjan(x.u,0);
        if (!dfn[x.v]) tarjan(x.v,0);
    }//tarjan
    vector<node> tmp;
    int all=0;
    for (auto x:nodes) {
```

```
    if (id[x.u]!=id[x.v]) {
        node nxt=x;
        nxt.u=id[x.u]; nxt.v=id[x.v];
        tmp.push_back(nxt);
        if (x.l<=l&&r<=x.r) all+=x.w;
    }
}//init
nodes.swap(tmp);
if (l==r) {
    ans[l]=base+all;
    return;
} int mid=(l+r)/2,div;

tmp.clear();
clear_and_set(l,r,nodes);
for (auto x:nodes) key[x.u]=key[x.v]=0;
for (auto x:nodes) {
    if (x.l<=mid&&!(x.l<=l&&r<=x.r)) {
        key[x.u]=key[x.v]=1;
        tmp.push_back(x);
    }
} div=0; remain.clear();
for (auto x:nodes) {
    if (x.l<=mid&&!(x.l<=l&&r<=x.r)) {
        if (!dfn[x.u]) build(x.u,0);
        if (!dfn[x.v]) build(x.v,0);
    }
}//tarjan
for (auto x:remain) tmp.push_back(x),div+=x.w;;
solve(l,mid,tmp,all-div+base);

tmp.clear();
clear_and_set(l,r,nodes);
for (auto x:nodes) key[x.u]=key[x.v]=0;
for (auto x:nodes) {
    if (mid<x.r&&!(x.l<=l&&r<=x.r)) {
        key[x.u]=key[x.v]=1;
        tmp.push_back(x);
    }
} div=0; remain.clear();
for (auto x:nodes) {
    if (mid<x.r&&!(x.l<=l&&r<=x.r)) {
        if (!dfn[x.u]) build(x.u,0);
        if (!dfn[x.v]) build(x.v,0);
    }
```

```
    }//tarjan
    for (auto x:remain) tmp.push_back(x),div+=x.w;;
    solve(mid+1,r,tmp,all-div+base);
}
int main() {
    int i,n;
    scanf("%d%d",&n,&m);
    map<pair<int,int>,int> MP;
    vector<node> init;
    FOR(i,1,m) {
        char op[4]; int u,v;
        scanf("%s%d%d",op,&u,&v);
        if (u>v) swap(u,v);
        if (op[0]=='A') {
            MP[make_pair(u,v)]=i;
        } else {
init.push_back(node(u,v,1,MP[make_pair(u,v)],i-1));
            MP.erase(make_pair(u,v));
        }
    } for (auto now:MP)
init.push_back(node(now.first.first,now.first.second,1,now.second,
m));
    solve(1,m,init,0);
    FOR(i,1,m) printf("%d\n",ans[i]);
    return 0;
}
```

二维树状数组

```
//poj2155,修改区间 01,query 单点 01,差分来做
int n,m;
int c[maxn][maxn];
int lowbit(int x){return x&-x;}
void update(int _x,int _y){
    for (int x=_x;x<=n;x+=lowbit(x))
        for (int y=_y;y<=n;y+=lowbit(y)) c[x][y]^=1;
}
int sum(int _x,int _y){
    int ret=0;
    for (int x=_x;x;x-=lowbit(x))
        for (int y=_y;y;y-=lowbit(y)) ret^=c[x][y];
    return ret;
}
```

树状数组 不大于 k 的最大值

```
const int MAX=1000000;
inline int lowbit(int x) {return x&-x;}
inline void insert(int x) {
    for (; x<=MAX; x+=lowbit(x)) a[x]++;
}
inline int find(int x) {
    while (x&&!a[x]) x^=lowbit(x);
    if (!x) return 0;
    int t=lowbit(x)>>1,y=a[x];
    while (t) {
        if (y-a[x-t]) y-=a[x-t];
        else {y=a[x-t]; x=x-t;}
        t>>=1;
    }
    return x;
}
```

BIT_差分

```
LL A[maxn],B[maxn];//A*i+B
inline int lowbit(int x){return x&-x;}
void Add(int x,LL val,LL VAL){
    for (;x<=n;x+=lowbit(x))
(A[x]+=val)%=M,(B[x]+=VAL)%=M;
}
void add(int l,int r,LL val){
    Add(l,val,-((l-1)*val%M)+M);
    Add(r+1,M-val,r*val%M);
}
LL query(int x){
    LL ret=0;for (int i=x;x;x-=lowbit(x))
(ret+=A[x]*i+B[x])%=M;
    return ret;
}
LL query(int l,int r){
    return (query(r)-query(l-1)+M)%M;
}
```

二维线段树

```
//单点修改区间查询 min,max
struct node{
    int left,right;
}treeX[maxn*4],treeY[maxn*4];
int a[maxn*4][maxn*4];
int mx[maxn*4][maxn*4],mn[maxn*4][maxn*4];
void buildY(int x,int y,int yl,int yr){
    treeY[y].left=yl,treeY[y].right=yr;
    if (yl==yr){
        if (treeX[x].left==treeX[x].right)
            mx[x][y]=mn[x][y]=a[treeX[x].left][yl];
        else{
            mx[x][y]=max(mx[x<<1][y],mx[x<<1|1][y]);
            mn[x][y]=min(mn[x<<1][y],mn[x<<1|1][y]);
        }
        return;
    }
    int mid=(yl+yr)/2;
    buildY(x,y<<1,yl,mid);
    buildY(x,y<<1|1,mid+1,yr);
    mx[x][y]=max(mx[x][y<<1],mx[x][y<<1|1]);
    mn[x][y]=min(mn[x][y<<1],mn[x][y<<1|1]);
}
void buildX(int x,int n,int xl,int xr){
    treeX[x].left=xl,treeX[x].right=xr;
    if (xl==xr){
        buildY(x,1,1,n);
        return;
    }
    int mid=(xl+xr)/2;
    buildX(x<<1,n,xl,mid);
    buildX(x<<1|1,n,mid+1,xr);
    buildY(x,1,1,n);
}

int querymaxY(int x,int y,int yl,int yr){
    int L=treeY[y].left,R=treeY[y].right;
    if (yl<=L&&R<=yr){
        return mx[x][y];
    }
    int mid=(L+R)/2,ret=0;
    if (mid>=yl) ret=max(ret,querymaxY(x,y<<1,yl,yr));
    if (yr>mid) ret=max(ret,querymaxY(x,y<<1|1,yl,yr));
    return ret;
}
int querymaxX(int x,int xl,int xr,int yl,int yr){
    int L=treeX[x].left,R=treeX[x].right;
    if (xl<=L&&R<=xr){
        return querymaxY(x,1,yl,yr);
    }
    int mid=(L+R)/2,ret=0;
    if (mid>=xl) ret=max(ret,querymaxX(x<<1,xl,xr,yl,yr));
    if (xr>mid) ret=max(ret,querymaxX(x<<1|1,xl,xr,yl,yr));
    return ret;
}

int queryminY(int x,int y,int yl,int yr){
    int L=treeY[y].left,R=treeY[y].right;
    if (yl<=L&&R<=yr){
        return mn[x][y];
    }
    int mid=(L+R)/2,ret=INF;
    if (mid>=yl) ret=min(ret,queryminY(x,y<<1,yl,yr));
    if (yr>mid) ret=min(ret,queryminY(x,y<<1|1,yl,yr));
    return ret;
}
int queryminX(int x,int xl,int xr,int yl,int yr){
    int L=treeX[x].left,R=treeX[x].right;
    if (xl<=L&&R<=xr){
        return queryminY(x,1,yl,yr);
    }
    int mid=(L+R)/2,ret=INF;
    if (mid>=xl) ret=min(ret,queryminX(x<<1,xl,xr,yl,yr));
    if (xr>mid) ret=min(ret,queryminX(x<<1|1,xl,xr,yl,yr));
    return ret;
}

void updateY(int x,int y,int posy,int val){
    int L=treeY[y].left,R=treeY[y].right;
    if (L==R){
        if (treeX[x].left==treeX[x].right)
            mx[x][y]=mn[x][y]=val;
        else{
            mx[x][y]=max(mx[x<<1][y],mx[x<<1|1][y]);
            mn[x][y]=min(mn[x<<1][y],mn[x<<1|1][y]);
        }
        return;
    }
    int mid=(L+R)/2;
    if (mid>=posy) updateY(x,y<<1,posy,val);
    else updateY(x,y<<1|1,posy,val);
    mx[x][y]=max(mx[x][y<<1],mx[x][y<<1|1]);
    mn[x][y]=min(mn[x][y<<1],mn[x][y<<1|1]);
}
void updateX(int x,int posx,int posy,int val){
    int L=treeX[x].left,R=treeX[x].right;
    if (L==R){
        updateY(x,1,posy,val);
        return;
    }
    int mid=(L+R)/2;
    if (mid>=posx) updateX(x<<1,posx,posy,val);
    else updateX(x<<1|1,posx,posy,val);
    updateY(x,1,posy,val);
}
int n,m,q;
int i,j;
int ans;
int main(){
    int T,x=0;
    scanf("%d",&T);
    while (T--){
        scanf("%d",&n);
        FOR(i,1,n)
            FOR(j,1,n) scanf("%d",&a[i][j]);
        buildX(1,n,1,n);
        scanf("%d",&q);
        printf("Case #%d:\n",++x);
        while (q--){
            int x,y,r;
            scanf("%d%d%d",&x,&y,&r);
            r/=2;
            int xl=max(1,x-r),xr=min(n,x+r);
            int yl=max(1,y-r),yr=min(n,y+r);
            int MX=querymaxX(1,xl,xr,yl,yr);
            int MN=queryminX(1,xl,xr,yl,yr);
            updateX(1,x,y,(MX+MN)/2);
            printf("%d\n",(MX+MN)/2);
        }
    }
}
```

吉爷爷线段树

```
// 区间取 max min，维护其他值
// 直接维护 max,min,第二小即可，暴力改就行了，一个 log
struct node{
    int MIN,MINCNT,IMIN;
```

```
        ll SUM;
        node(){MIN=MINCNT=SUM=0; IMIN=INF;}
    }T[maxn*4];
    void min_(int &A,int B){(A>B)&&(A=B);}
    char op[2]; ll ans;
    void build(int x,int l,int r){
        T[x]=node(); T[x].MINCNT=r-l+1;
        if (l==r) return;
        int mid=(l+r)/2;
        build(x<<1,l,mid);
        build(x<<1|1,mid+1,r);
    }
    inline void update(int x,int l,int r,int val,int L,int R,int tag=1)
{
        if (l<=L&&R<=r){
            if (T[x].IMIN>val){
                if (T[x].MIN<val) {
                    T[x].SUM+=(ll)(val-T[x].MIN)*T[x].MINCNT;
                    T[x].MIN=val;
                } if (tag) ans+=T[x].SUM;
                return;
            }
        } int mid=(L+R)/2;
        update(x<<1,L,mid,T[x].MIN,L,mid,0);
        update(x<<1|1,mid+1,R,T[x].MIN,mid+1,R,0);
        if (l<=mid) update(x<<1,l,r,val,L,mid);
        if (mid<r) update(x<<1|1,l,r,val,mid+1,R);
        T[x].SUM=T[x<<1].SUM+T[x<<1|1].SUM;
        T[x].MIN=T[x].IMIN=INF;
        T[x].MIN=min(T[x<<1].MIN,T[x<<1|1].MIN);
        T[x].MINCNT=0;

        if (T[x<<1].MIN==T[x].MIN) {
            min_(T[x].IMIN,T[x<<1].IMIN);
            T[x].MINCNT+=T[x<<1].MINCNT;
        } else min_(T[x].IMIN,T[x<<1].MIN);
        if (T[x<<1|1].MIN==T[x].MIN) {
            min_(T[x].IMIN,T[x<<1|1].IMIN);
            T[x].MINCNT+=T[x<<1|1].MINCNT;
        } else min_(T[x].IMIN,T[x<<1|1].MIN);
    }
```

扫描线 矩形周长并

```
    int size;
    int len[maxn*2];
    int n,m,i,j,k;
    struct Seg {
        struct node {
            int left,right;
            int len,num;
            bool cl,cr;//iff
            int lazy;
            void update(int x) {
                lazy+=x;
            }
        } tree[maxn*4];
        void pushup(int x) {
            if (tree[x].lazy) {
                tree[x].len=len[tree[x].right+1]-len[tree[x].left];
                tree[x].cl=tree[x].cr=1; tree[x].num=2;
            } else if (tree[x].left==tree[x].right) {
                tree[x].len=0;
                tree[x].cl=tree[x].cr=0; tree[x].num=0;
            } else {
                tree[x].len=tree[x<<1].len+tree[x<<1|1].len;
                tree[x].num=tree[x<<1].num+tree[x<<1|1].num;
                if (tree[x<<1].cr&&tree[x<<1|1].cl)
tree[x].num-=2;
                tree[x].cl=tree[x<<1].cl;
                tree[x].cr=tree[x<<1|1].cr;
            }
        };
        void build(int x,int l,int r) {
            tree[x].left=l; tree[x].right=r;
            tree[x].len=tree[x].lazy=0;
            if (l==r) {
            } else {
                int mid=(l+r)/2;
                build(x<<1,l,mid);
                build(x<<1|1,mid+1,r);
                pushup(x);
            }
        }
        void update(int x,int l,int r,LL val) {
            int L=tree[x].left,R=tree[x].right;
            if (l<=L&&R<=r) {
                tree[x].update(val);
                pushup(x);
            } else {
```

```
            int mid=(L+R)/2;
            if (mid>=l) update(x<<1,l,r,val);
            if (r>mid) update(x<<1|1,l,r,val);
            pushup(x);
        }
    }
    int query(int x,int l,int r) { //num
        int L=tree[x].left,R=tree[x].right;
        if (l<=L&&R<=r) {
            return tree[x].len;
        } else {
            int mid=(L+R)/2;
            int ans;
            if (mid>=l) ans+=query(x<<1,l,r);
            if (r>mid) ans+=query(x<<1|1,l,r);
            pushup(x);
            return ans;
        }
    }
} T;
struct point {
    int x1,x2,h;
    int n;
    bool operator <(const point &a)const {
        if (h!=a.h) return h<a.h;
        return n>a.n;
    }
} a[maxn];
map<int,int> Hash;
int x1,x2,y1,y2;
int ans;
int len1,len2,num;
int main() {
    while (~scanf("%d",&n)) {
        if (n==0) break;
        FOR(i,1,n) {
            scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            len[i*2-1]=x1; len[i*2]=x2;
            a[i*2-1].x1=x1; a[i*2-1].x2=x2;
            a[i*2-1].n=1;   a[i*2-1].h=y1;
            a[i*2].x1=x1; a[i*2].x2=x2;
            a[i*2].n=-1; a[i*2].h=y2;
        }
        sort(a+1,a+n*2+1);
        sort(len+1,len+n*2+1);
        Hash.clear();
        FOR(i,1,2*n) Hash[len[i]]=i;
        T.build(1,1,n*2);
        ans=0;
        FOR(i,1,2*n) {
            len1=T.tree[1].len; num=T.tree[1].num;
            T.update(1,Hash[a[i].x1],Hash[a[i].x2]-1,a[i].n);
            len2=T.tree[1].len;
            ans+=abs(len2-len1);
            ans+=num*(a[i].h-a[i-1].h);
        }
        printf("%d\n",ans);
    }
}
```

主席树

//静态区间第 k 大

```
vector<int> v;//学到的 hash 方法
int getid(int x){return
lower_bound(v.begin(),v.end(),x)-v.begin()+1;}
int root[maxn],a[maxn],cnt;
struct Tnode{
    int left,right,sum;
}T[maxn*40];
void update(int l,int r,int &x,int y,int pos){
    T[++cnt]=T[y];T[cnt].sum++;x=cnt;
    if (l==r) return;
    int mid=(l+r)/2;
    if (mid>=pos) update(l,mid,T[x].left,T[y].left,pos);
    else update(mid+1,r,T[x].right,T[y].right,pos);
}
int query(int l,int r,int x,int y,int k){
    if (l==r) return l;
    int mid=(l+r)/2;
    int sum=T[T[y].left].sum-T[T[x].left].sum;
    if (sum>=k) return query(l,mid,T[x].left,T[y].left,k);
    else return query(mid+1,r,T[x].right,T[y].right,k-sum);
}
```

可持久化数组(主席树维护)

```
struct Tnode {
    int left,right,val;
} T[maxn*80];
```

```
int cnt=0;
void build(int &x,int l,int r) {
    if (!x) x=++cnt;
    if (l==r) {T[x].val=l; return;}
    int mid=(l+r)/2;
    build(T[x].left,l,mid);
    build(T[x].right,mid+1,r);
}
void update(int &x,int y,int pos,int val,int l,int r) {
    T[++cnt]=T[y]; x=cnt;
    if (l==r) {T[x].val=val; return;}
    int mid=(l+r)/2;
    if (mid>=pos) update(T[x].left,T[y].left,pos,val,l,mid);
    else update(T[x].right,T[y].right,pos,val,mid+1,r);
}
int query(int x,int pos,int l,int r) {
    if (l==r) return T[x].val;
    int mid=(l+r)/2;
    if (mid>=pos) return query(T[x].left,pos,l,mid);
    else return query(T[x].right,pos,mid+1,r);
}
int root[maxn];
int n,m;
int i,j,k,t;
int a,b,ans;
inline int getfather(int x) {
    int t=query(root[i],x,1,n);
    if (t==x) return x;
    int fa=getfather(t);
    update(root[i],root[i],x,fa,1,n);
    return fa;
}
int main() {
    scanf("%d%d",&n,&m);
    build(root[0],1,n);
    FOR(i,1,m) {
        scanf("%d",&k);
        root[i]=root[i-1];
        if (k==1) {
            scanf("%d%d",&a,&b);
            a^=ans; b^=ans;
            int x=getfather(a),y=getfather(b);
            if (x==y) continue;
            update(root[i],root[i],x,y,1,n);
        } else if (k==2) {
            scanf("%d",&t);
            t^=ans;
            root[i]=root[t];
        } else {
            scanf("%d%d",&a,&b);
            int x=getfather(a),y=getfather(b);
            a^=ans; b^=ans;
            if (x==y) puts("1"),ans=1;
            else puts("0"),ans=0;
        }
    }
}
```

树套树

// zoj2112 动态第 k 大(这个是类似 kuangbin 大佬的做法按点建树，我按权值多个 log...)

```
struct node{
    int l,r,cnt;
    node(){l=r=cnt=0;}
}T[2500010];
int cnt;
int SIZE;
inline int lowbit(int x){return x&(-x);}
void Update(int &x,int y,int l,int r,int pos,int val){
    T[++cnt]=T[y];T[cnt].cnt+=val;x=cnt;
    if (l==r) return;
    int mid=(l+r)/2;
    if (mid>=pos) Update(T[x].l,T[y].l,l,mid,pos,val);
    else Update(T[x].r,T[y].r,mid+1,r,pos,val);
}
int n,m;
int root[maxn];
void update(int x,int pos,int val){
    while (x<=n){
        Update(root[x],root[x],1,SIZE,pos,val);
        x+=lowbit(x);
    }
}
int ROOT[maxn];
int useL[maxn],useR[maxn];//现在的 l/r
int Query(int l,int r,int L,int R,int pos,int pre_L,int pre_R){//颜色,pos L->R
    if (l==r) return l;
    int x;
```

```
        int mid=(l+r)/2,nowcnt=0;
        for(x=L-1;x;x-=lowbit(x)) nowcnt-=T[T[useL[x]].l].cnt;
        for(x=R;x;x-=lowbit(x))    nowcnt+=T[T[useR[x]].l].cnt;
        nowcnt+=T[T[pre_R].l].cnt-T[T[pre_L].l].cnt;
        if (nowcnt>=pos){
            for(x=L-1;x;x-=lowbit(x)) useL[x]=T[useL[x]].l;
            for(x=R;x;x-=lowbit(x))    useR[x]=T[useR[x]].l;
            return Query(l,mid,L,R,pos,T[pre_L].l,T[pre_R].l);
        }else{
            for(x=L-1;x;x-=lowbit(x)) useL[x]=T[useL[x]].r;
            for(x=R;x;x-=lowbit(x))    useR[x]=T[useR[x]].r;
            return
Query(mid+1,r,L,R,pos-nowcnt,T[pre_L].r,T[pre_R].r);
        }
    }
    int query(int L,int R,int pos){
        int x;
        for(x=L-1;x;x-=lowbit(x)) useL[x]=root[x];
        for(x=R;x;x-=lowbit(x))    useR[x]=root[x];
        return Query(1,SIZE,L,R,pos,ROOT[L-1],ROOT[R]);
    }
    char K[maxn],Q[20];
    int A[maxn][4];
    int a[maxn];
    vector<int> H;
    inline int getid(int x){return
lower_bound(H.begin(),H.end(),x)-H.begin()+1;}
    void solve(){
        scanf("%d%d",&n,&m);
        int i;
        FOR(i,1,n) scanf("%d",&a[i]),H.push_back(a[i]);
        REP(i,m){
            scanf("%s",Q);
            K[i]=Q[0];
            if (K[i]=='Q')
scanf("%d%d%d",&A[i][0],&A[i][1],&A[i][2]);
            if (K[i]=='C')
scanf("%d%d",&A[i][0],&A[i][1]),H.push_back(A[i][1]);
        }

sort(H.begin(),H.end());H.erase(unique(H.begin(),H.end()),H.end());
        SIZE=H.size();
        cnt=0;
        FOR(i,1,n)
Update(ROOT[i],ROOT[i-1],1,SIZE,getid(a[i]),1);
        REP(i,m){
            if (K[i]=='Q')
printf("%d\n",H[query(A[i][0],A[i][1],A[i][2])-1]);//l,r,pos
            if (K[i]=='C'){
                update(A[i][0],getid(a[A[i][0]]),-1);
                a[A[i][0]]=A[i][1];
                update(A[i][0],getid(A[i][1]),1);
            }
        }
        FOR(i,1,n) root[i]=0;
        FOR(i,1,cnt) T[i]=node();
        vector<int>().swap(H);
    }
```

CDQ 分治(套线段树)

```
// CF848C CDQ 分治（区间数字出现的 r-l 之和）
//将所有操作计算成为 add 和 del,然后 solve(l,r),再去除影响
const LL MAX=10000007;
struct node{
    int l,r; LL sum;
}T[MAX];
int cnt;
void Update(int &x,int pos,int val,int l,int r){
    if (!x) x=++cnt;
    T[x].sum+=val;
    if (l==r) return;
    int mid=(l+r)/2;
    if (mid>=pos) Update(T[x].l,pos,val,l,mid);
    else Update(T[x].r,pos,val,mid+1,r);
}
LL Query(int x,int l,int r,int L,int R){
    if (!x||(l<=L&&R<=r)) return T[x].sum;
    int mid=(L+R)/2;
    LL ret=0;
    if (mid>=l) ret+=Query(T[x].l,l,r,L,mid);
    if (r>mid) ret+=Query(T[x].r,l,r,mid+1,R);
    return ret;
}
int n,m;
int root[maxn];
inline int lowbit(int x){
    return x&-x;
}
void update(int x,int pos,int val){
    for (;x<=n;x+=lowbit(x)) Update(root[x],pos,val,1,n);
}
LL query(int x,int l,int r){
    LL ret=0;
    for (;x;x-=lowbit(x))
        ret+=Query(root[x],l,r,1,n);//其实还是应该是 r-(l-1)的
    return ret;
}
int a[maxn];
set<int> S[maxn];
void ins(int pos,int val){//固定 R (L 用前缀和)
    S[val].insert(pos);
    set<int>::iterator it=S[val].lower_bound(pos),itt=it;itt++;
    int pre=0,suf=0;
    if (it!=S[val].begin()) it--,pre=*it;
    if (itt!=S[val].end()) suf=*itt;
    if (pre) update(pos,pre,pos-pre);
    if (suf) update(suf,pos,suf-pos);
    if (pre&&suf) update(suf,pre,pre-suf);
}
void del(int pos,int val){
    set<int>::iterator it=S[val].lower_bound(pos),itt=it;itt++;
    int pre=0,suf=0;
    if (it!=S[val].begin()) it--,pre=*it;
    if (itt!=S[val].end()) suf=*itt;
    if (pre) update(pos,pre,-(pos-pre));
    if (suf) update(suf,pos,-(suf-pos));
    if (pre&&suf) update(suf,pre,-(pre-suf));
    S[val].erase(pos);
}
int i;
int main(){
    scanf("%d%d",&n,&m);
    FOR(i,1,n){
        scanf("%d",&a[i]);
        ins(i,a[i]);
    }
    REP(i,m){
        int k;
        scanf("%d",&k);
        if (k==1){
            int p,x;
            scanf("%d%d",&p,&x);
            del(p,a[p]);
            a[p]=x;
            ins(p,a[p]);
        }else if (k==2){
            int l,r;
            scanf("%d%d",&l,&r);
            printf("%I64d\n",query(r,l,r));
        }
    }
}
```

SPLAY

```
int A[maxn];
struct splay_tree {
    struct node {
        int val,min,max,add,size,son[2];//add=lazy
        bool rev;
        void init(int _val) { //开始时 T[i].val==a[i-1](线性的);
            val=min=max=_val; size=1;
            if (_val==INF) max=-INF;
            add=son[0]=son[1]=0; rev=0;
        }
    } T[maxn*2]; //内存池
    int fa[maxn*2],root,tot;
    void pushup(int x) {
        T[x].min=T[x].max=T[x].val; T[x].size=1;
        if (T[x].val==INF) T[x].max=-INF;
        if (T[x].son[0]) {
            T[x].min=min(T[x].min,T[T[x].son[0]].min);
            T[x].max=max(T[x].max,T[T[x].son[0]].max);
            T[x].size+=T[T[x].son[0]].size;
        }
        if (T[x].son[1]) {
            T[x].min=min(T[x].min,T[T[x].son[1]].min);
            T[x].max=max(T[x].max,T[T[x].son[1]].max);
            T[x].size+=T[T[x].son[1]].size;
        }
    }
    void pushdown(int x) {
        if (x==0) return;
        if (T[x].add) {
            if (T[x].son[0]) {
                T[T[x].son[0]].val+=T[x].add;
                T[T[x].son[0]].min+=T[x].add;
                T[T[x].son[0]].max+=T[x].add;
                T[T[x].son[0]].add+=T[x].add;
            }
            if (T[x].son[1]) {
                T[T[x].son[1]].val+=T[x].add;
                T[T[x].son[1]].min+=T[x].add;
                T[T[x].son[1]].max+=T[x].add;
                T[T[x].son[1]].add+=T[x].add;
            }
            T[x].add=0;
        }
        if (T[x].rev) {
            if (T[x].son[0]) T[T[x].son[0]].rev^=1;
            if (T[x].son[1]) T[T[x].son[1]].rev^=1;
            swap(T[x].son[0],T[x].son[1]);
            T[x].rev=0;
        }
    }
    void rotate(int x,int kind) { //zig(1->) zag(0<-)都行
        int y=fa[x],z=fa[y];
        T[y].son[!kind]=T[x].son[kind],fa[T[x].son[kind]]=y;
        T[x].son[kind]=y,fa[y]=x;
        T[z].son[T[z].son[1]==y]=x,fa[x]=z;
        pushup(y);
    }
    void splay(int x,int goal) { //node x->goal's son
        if (x==goal) return;
        while (fa[x]!=goal) {
            int y=fa[x],z=fa[y];
            pushdown(z),pushdown(y),pushdown(x);
            int rx=T[y].son[0]==x,ry=T[z].son[0]==y;
            if (z==goal) rotate(x,rx);
            else {
                if (rx==ry) rotate(y,ry);
                else rotate(x,rx);
                rotate(x,ry);
            }
        } pushup(x);
        if (goal==0) root=x;
    }
    int select(int pos) { //getnode
        int u=root;
        pushdown(u);
        while (T[T[u].son[0]].size!=pos) { //这里由于头节点有个
-INF 所以不-1
            if (pos<T[T[u].son[0]].size) u=T[u].son[0];
            else {
                pos-=T[T[u].son[0]].size+1;
                u=T[u].son[1];
            } pushdown(u);
        } return u;
    }
//下面是自己写的一点常用?函数
    void update(int l,int r,int val) {
        int u=select(l-1),v=select(r+1);
        splay(u,0); splay(v,u);
        T[T[v].son[0]].min+=val;
        T[T[v].son[0]].max+=val;
        T[T[v].son[0]].val+=val;
        T[T[v].son[0]].add+=val;//lazy
    }
    void reverse(int l,int r) {
        int u=select(l-1),v=select(r+1);
        splay(u,0); splay(v,u);
        T[T[v].son[0]].rev^=1;
    }
    void revolve(int l,int r,int x) { //l~r->循环往后 x 位
        int u=select(r-x),v=select(r+1);
        splay(u,0); splay(v,u);
        int tmp=T[v].son[0]; T[v].son[0]=0;
        pushup(v); pushup(u);
        u=select(l-1),v=select(l);
        splay(u,0); splay(v,u);
        fa[tmp]=v; T[v].son[0]=tmp;
        pushup(v); pushup(u);
    }
    void cut(int l,int r,int x) { //l~r->去掉的 x 位置后
//HDU3487
        int u=select(l-1),v=select(r+1);
        splay(u,0); splay(v,u);
        int tmp=T[v].son[0];
        T[v].son[0]=0;
        pushup(v); pushup(u);
        u=select(x); v=select(x+1);
        splay(u,0); splay(v,u);
        fa[tmp]=v; T[v].son[0]=tmp;
        pushup(v); pushup(u);
    }
    int query_min(int l,int r) {
        int u=select(l-1),v=select(r+1);
        splay(u,0); splay(v,u);
        return T[T[v].son[0]].min;
    }

    void insert(int x,int val) {
        int u=select(x),v=select(x+1);
        splay(u,0); splay(v,u);
        ++tot; if (tot==maxn) tot=1;
        T[tot].init(val); fa[tot]=v;
        T[v].son[0]=tot;
        pushup(v); pushup(u);
    }
    void delfree(int x) {//buffer
        if (x==0) return;
        bufs++; if (bufs==maxn) bufs=1;
        nodebuff[bufs]=x;
        delfree(T[x].son[0]);
        delfree(T[x].son[1]);
    }
    void erase(int l,int r) {
        int u=select(l-1),v=select(r+1);
        splay(u,0); splay(v,u);
        delfree(T[v].son[0]);
        T[v].son[0]=0;
        pushup(v); pushup(u);
    }
    void exchange(int l1,int r1,int l2,int r2)
{ //r1-l1+1?=r2-l2+1   OK
        if (l1>l2) {swap(l1,l2); swap(r1,r2);}
        int u=select(l1-1),v=select(r1+1);
        splay(u,0); splay(v,u);
        int tmp=T[v].son[0]; T[v].son[0]=0;
        pushup(v); pushup(u);
        l2-=T[tmp].size; r2-=T[tmp].size;
        int _u=select(l2-1),_v=select(r2+1);
        splay(_u,0); splay(_v,_u);
        fa[tmp]=_v;
        swap(T[_v].son[0],tmp);
        pushup(_v); pushup(_u);
        u=select(l1-1),v=select(l1);
        splay(u,0); splay(v,u);
        fa[tmp]=v;
        T[v].son[0]=tmp;
        pushup(v); pushup(u);
    }

    int nodebuff[maxn],bufs;//bufs:position
    int build(int l,int r) { //add_list
        if (l>r) return 0;
        ++tot; if (tot==maxn) tot=1;
        int ret=nodebuff[tot];
        int mid=(l+r)/2;
        T[ret].init(A[mid]);
        if (l==r) return ret;
        int ls=build(l,mid-1);
        int rs=build(mid+1,r);
        if (ls) fa[ls]=ret,T[ret].son[0]=ls;
        if (rs) fa[rs]=ret,T[ret].son[1]=rs;
        pushup(ret);
        return ret;
    }
    void init(int n) {
        int i; tot=0;
        REP(i,maxn) nodebuff[i]=i;
        rFOR(i,1,n) A[i+1]=A[i];
        A[1]=A[n+2]=-INF;
        root=build(1,n+2);
        fa[root]=0; T[0].init(-INF);
        fa[0]=0; T[0].son[1]=root; T[0].size=0;
    }
} T;
```

SPLAY 启发式合并

```
//HDU6133，一棵树的合并
struct splaytree {
    struct node {
        LL val,sum;
        int son[2],size;
        void init(LL _val) {
            val=sum=_val; size=1;
            son[0]=son[1]=0;
        }
    } T[maxn]; //编号是对应的
    int fa[maxn];
    int root;
    inline void pushup(int x) {
        T[x].sum=T[x].val;
        T[x].size=1;
        if (T[x].son[0]) {
            T[x].sum+=T[T[x].son[0]].sum;
            T[x].size+=T[T[x].son[0]].size;
        }
        if (T[x].son[1]) {
            T[x].sum+=T[T[x].son[1]].sum;
            T[x].size+=T[T[x].son[1]].size;
        }
    }
    void rotate(int x,int kind) {
        int y=fa[x],z=fa[y];
        T[y].son[!kind]=T[x].son[kind],fa[T[x].son[kind]]=y;
        T[x].son[kind]=y,fa[y]=x;
        T[z].son[T[z].son[1]==y]=x,fa[x]=z;
        pushup(y);
    }
    void splay(int x,int goal) {
        if (x==goal) return;
        while (fa[x]!=goal) {
            int y=fa[x],z=fa[y];
            int rx=T[y].son[0]==x,ry=T[z].son[0]==y;
            if (z==goal) rotate(x,rx);
            else {
                if (rx==ry) rotate(y,ry);
                else rotate(x,rx);
                rotate(x,ry);
            }
        }
        pushup(x);
        if (goal==0) root=x;
    }
    LL insert(int x) { //x 为原先位置
        int u=root,f=0;
        while (u) {
            f=u;
            if (T[x].val<T[u].val) u=T[u].son[0];
            else u=T[u].son[1];
        }
        if (T[x].val<T[f].val) T[f].son[0]=x;
        else T[f].son[1]=x;
        fa[x]=f;
        splay(x,0);
        return
T[T[x].son[0]].sum+T[x].val*(T[T[x].son[1]].size+1);
    }
    LL dfs(int x) {
        int l=T[x].son[0],r=T[x].son[1];
        LL ret=0;
        T[x].init(T[x].val);
        if (l) ret+=dfs(l);
        ret+=insert(x);
        if (r) ret+=dfs(r);
        return ret;
    }
    LL merge(int x,int y,LL tmp,LL ret) {
        if (x==y) return tmp;
        splay(x,0); splay(y,0);
        if (T[x].size>T[y].size) swap(x,y),swap(tmp,ret);
        root=y;
        ret+=dfs(x);
        return ret;
    }
    int getkth(int x,int k) { //未验证,抄的前面那个板子
        int u=root;
        while (T[T[u].son[0]].size!=k) {
            if (k<T[T[u].son[0]].size) u=T[u].son[0];
            else {
                k-=T[T[u].son[0]].size+1;
                u=T[u].son[1];
            }
        }
        return T[x].val;
    }
} T;
```

左偏树

```
struct node{
    int l,r,val,len;
    node(int _val=0){l=r=len=0; val=_val;}
}T[maxn]; int tot;
int merge(int x,int y){//不能直接 swap x 和儿子，否则可能不满足
堆性质
    if (!x||!y) return x|y;
    if (T[x].val>T[y].val) swap(x,y);
    T[x].r=merge(T[x].r,y);
    if (T[T[x].l].len<T[T[x].r].len) swap(T[x].l,T[x].r);
    T[x].len=T[T[x].r].len+1;
    return x;
}
int pop(int x) {
    T[x].val=-1;
    return merge(T[x].l,T[x].r);
}
```

可持久化非旋 treap

```
// 内存回收: 没写, 写的话可以直接用个东西保存指向它的 pointer 个数
// 辣鸡蓝桥杯的题目, 喵的什么垃圾评测机, 全 MLE 是个什么东西
namespace persist_treap {
    typedef pair<int,int> Pair;
    const int maxn=1e7;//maxn>=2
    struct node {
        int l,r,len,size;
        ll val,lazy,sum;
        node(ll _val=0) {
            l=r=len=0; val=_val; lazy=0;
            sum=val; size=1;
        }
    } T[maxn];
    int root;
    int pool[maxn],st,ed;//ends
    void init() {
        int i; ed=maxn-1; root=0; T[0].size=0;//0:no use
        REP(i,maxn-1) pool[i]=i+1;//start from 1
    }
    void delnode(int pos) {
        if (ed==maxn-1) ed=0;
        // T[pos]=node();// no use
        pool[ed++]=pos;
    }
    int insnode(ll x) {//value
        // assert(st+1!=ed);// no !!!
        int pos=pool[st++];
        if (st==maxn-1) st=0;
        T[pos]=node(x);
        return pos;
    }
    int persistnode(int ini) {
        // assert(st+1!=ed);// no !!!
        int pos=pool[st++];
        T[pos]=T[ini];
        if (st==maxn-1) st=0;
        return pos;
    }
    void ADD(int x,ll val) { //update
        T[x].lazy+=val; T[x].val+=val;
        T[x].sum+=T[x].size*val;
    }
    bool pushdown(int x) {
        if (!T[x].lazy) return 0;
        if (T[x].l) {
            T[x].l=persistnode(T[x].l);
            ADD(T[x].l,T[x].lazy);
        }
        if (T[x].r) {
            T[x].r=persistnode(T[x].r);
            ADD(T[x].r,T[x].lazy);
        }
        T[x].lazy=0;
        return 1; // changed; -1/2 空间
    }
    void pushup(int x) {
        T[x].sum=T[x].val;
        T[x].len=0; T[x].size=1;
        if (T[x].l) {
            T[x].sum+=T[T[x].l].sum;
            T[x].len=max(T[x].len,T[T[x].l].len+1);
            T[x].size+=T[T[x].l].size;
        }
        if (T[x].r) {
            T[x].sum+=T[T[x].r].sum;
            T[x].len=max(T[x].len,T[T[x].r].len+1);
            T[x].size+=T[T[x].r].size;
        }
    }
    int merge(int x,int y,bool downx=0,bool downy=0) {
        if (!x||!y) return x|y;
        if (T[x].len>T[y].len) {
            if (!downx) x=persistnode(x);
            bool okay=pushdown(x);
            T[x].r=merge(T[x].r,y,okay,downy);
            pushup(x); return x;
        } else {
            if (!downy) y=persistnode(y);
            bool okay=pushdown(y);
            T[y].l=merge(x,T[y].l,downx,okay);
            pushup(y); return y;
        }
    }
    pii split(int x,int k,bool down=0) {
        if (!x) return make_pair(0,0);
        if (!down) x=persistnode(x);
        bool persisted=pushdown(x); //persist:newnode
        // printf("split: %lld; sz=%d;
k=%d\n",T[x].val,T[x].size,k);
        pii P;
        if (!k||T[T[x].l].size>=k) {
            // printf("to_left %d\n",T[x].l);
            P=split(T[x].l,k,persisted);
            T[x].l=P.second; pushup(x); P.second=x;
        } else {
            P=split(T[x].r,k-T[T[x].l].size-1,persisted);
            T[x].r=P.first; pushup(x); P.first=x;
        } return P;
    }
    void print_dfs(int x) {
        if (!x) return;
        print_dfs(T[x].l);
        printf("%lld ",T[x].val);
        print_dfs(T[x].r);
    }

    ll query(int l,int r) {//用个东西记录一下??
        pii A=split(root,l-1);
        // print_dfs(A.first); puts("A.first");
        pii B=split(A.second,r-l+1);
        // print_dfs(B.first); puts("B.first");
        // print_dfs(B.second); puts("B.second");
        return T[B.first].sum;
    }
    void update(int l,int r,ll val) {
        pii A=split(root,l-1);
        pii B=split(A.second,r-l+1);
        ADD(B.first,val);
        root=merge(merge(A.first,B.first),B.second);
    }
    void insert(int k,int val) {//after kth
        pii A=split(root,k);
        int y=insnode(val);
        // print_dfs(A.first); puts("okay");
        // printf("root=%d\n",root);
        root=merge(A.first,merge(y,A.second));
    }
    void transto(int l,int r,int x,int y) {
        pii A=split(root,l-1);
        pii B=split(A.second,r-l+1);
        pii A_=split(root,x-1);
        pii B_=split(A_.second,y-x+1);
        root=merge(merge(A.first,B_.first),B.second);
    }
}
```

LCT

```
//确认没写错，加边减边，改边权，查第二大值
//修改边权:把边当成点,mark 一下,然后左右端点连边即可
//hdu5002, chain_makeSame; query secondary_max
namespace LCT {
    const int maxn=1e5+7;
    struct info {
        int size;
        pii max1,max2;
        info(int _val=-INF,int _cnt=1,int _size=1):

size(_size),max1(make_pair(_val,_cnt)),max2(make_pair(-INF,0))
{}
        void print() {
            debug(" debug: infomation: max=(%d,%d)(%d,%d)
size=%d\n",max1.first,max1.second,max2.first,max2.second,size)
;
        }
    };
    struct tag {
        int same,add;//same:lazy
        tag() {same=-INF; add=0;}
        bool tagadd() {return (add!=0);}
        bool tagsame() {return (same!=-INF);}
    };
    //info_merge
    inline void merge(info &x,pii value) {
        if (x.max1.first==value.first)
            return (void)(x.max1.second+=value.second);
        if (x.max1<value) swap(x.max1,value);
        if (x.max2.first==value.first)
            return (void)(x.max2.second+=value.second);
        if (x.max2<value) swap(x.max2,value);
    }
    info merge(const info &x,const info &y) {
        info ret=x;
        ret.size+=y.size;
        merge(ret,y.max1);
    merge(ret,y.max2);
    return ret;
}
//info_update and tag_update
inline void MakeSame(info &_info,int value) {
    _info.max1=make_pair(value,_info.size);
    _info.max2=make_pair(-INF,0);
}
inline void MakeSame(tag &_tag,int value) {
    _tag.same=value;
}
inline void AddValue(info &_info,int value) {
    _info.max1.first+=value;
    if (_info.max2.first!=-INF) _info.max2.first+=value;
}
inline void AddValue(tag &_tag,int value) {
    _tag.add+=value;
    if (_tag.tagsame()) _tag.same+=value;
}
struct node {
    int son[2],fa;
    int val;
    info chain; tag chaintag;
    bool rev,isroot;//root=1:isroot
    void init(int _val) {
        val=_val;
        chain=info(val);
        chaintag=tag();
        rev=0; son[0]=son[1]=0;
        fa=0; isroot=1;
    }
} T[maxn];
void Reverse(int x) {
    T[x].rev^=1;
    swap(T[x].son[0],T[x].son[1]);
}
void AddValue(node &x,int val) {//Add_To_Node
    x.val+=val;
    AddValue(x.chain,val);
    AddValue(x.chaintag,val);
}
void MakeSame(node &x,int val) {
    x.val=val;
    MakeSame(x.chain,val);
    MakeSame(x.chaintag,val);
    }
    void pushup(int x) {
        T[x].chain=info(T[x].val);//clear
        if (T[x].son[0])

T[x].chain=merge(T[T[x].son[0]].chain,T[x].chain);
        if (T[x].son[1])

T[x].chain=merge(T[x].chain,T[T[x].son[1]].chain);
    }
    void pushdown(int x) {
        if (T[x].rev) {
            if (T[x].son[0]) Reverse(T[x].son[0]);
            if (T[x].son[1]) Reverse(T[x].son[1]);
            T[x].rev=0;
        }
        if (T[x].chaintag.tagadd()) {
            if (T[x].son[0])
AddValue(T[T[x].son[0]],T[x].chaintag.add);
            if (T[x].son[1])
AddValue(T[T[x].son[1]],T[x].chaintag.add);
            T[x].chaintag.add=0;
        }
        if (T[x].chaintag.tagsame()) {
            if (T[x].son[0])
MakeSame(T[T[x].son[0]],T[x].chaintag.same);
            if (T[x].son[1])
MakeSame(T[T[x].son[1]],T[x].chaintag.same);
            T[x].chaintag.same=-INF;
        }
    }
    void rotate(int x,int kind) {
        int y=T[x].fa,z=T[y].fa;
        T[y].son[!kind]=T[x].son[kind],T[T[x].son[kind]].fa=y;
        T[x].son[kind]=y,T[y].fa=x;
        if (T[y].isroot) {T[x].isroot=true; T[y].isroot=false;}
        else T[z].son[T[z].son[1]==y]=x;
        T[x].fa=z; pushup(y);
    }
    void PreChange(int x) {//change_from_root
        static int ids[maxn],i,k;
        for (k=0; !T[x].isroot; k++){
            ids[k]=x,x=T[x].fa;
        } ids[k++]=x;
        rREP(i,k) pushdown(ids[i]);
    }
    void splay(int x) { //to root
        PreChange(x);
        while (!T[x].isroot) {
            int y=T[x].fa,z=T[y].fa;
            int rx=T[y].son[0]==x,ry=T[z].son[0]==y;
            if (T[y].isroot) rotate(x,rx);
            else {
                if (rx==ry) rotate(y,ry);
                else rotate(x,rx);
                rotate(x,ry);
            }
        } pushup(x);
    }
    int access(int x) {
        int y=0;
        for (; x; x=T[x].fa) {
            splay(x);
            T[T[x].son[1]].isroot=true;
            T[x].son[1]=y;
            T[y].isroot=false;
            y=x; pushup(x);
        } return y;
    }
    bool judge(int u,int v) {
        while (T[u].fa) u=T[u].fa;
        while (T[v].fa) v=T[v].fa;
        return u==v;
    }
    void makeroot(int x) {
        access(x); splay(x);
        Reverse(x);
    }
    bool link(int u,int v) {
        if (judge(u,v)) return 1;
        makeroot(u); T[u].fa=v;
        return 0;
    }
    bool cut(int u,int v) {
        makeroot(u); splay(v);
        T[T[v].son[0]].fa=T[v].fa;
        T[v].fa=0;
        T[T[v].son[0]].isroot=true;
        T[v].son[0]=0;
        pushup(v);
        return 0;
    }
    bool add(int u,int v,int val) {
        makeroot(u); access(v); splay(v);
        AddValue(T[v],val);
        return 0;
    }
    bool change(int u,int v,int val) {
        makeroot(u); access(v); splay(v);
        MakeSame(T[v],val);
        return 0;
    }
    pair<int,int> ask(int u,int v) {
        makeroot(u); access(v); splay(v);
        return T[v].chain.max2;
    }
};
vector<int> edge[maxn];
void dfs(int x,int fa) {
    LCT::T[x].fa=fa;
    LCT::T[x].isroot=1;
    for (int v:edge[x]) if (v!=fa) dfs(v,x);
}
int main() {
    int x=0;
    int T,_; T=1;
    scanf("%d",&T);
    FOR(_,1,T) {
        int n,m,i;
        scanf("%d%d",&n,&m);
        FOR(i,1,n) {
            int val;
            scanf("%d",&val);
            LCT::T[i].init(val);
        }
        REP(i,n-1) {
            int u,v;
            scanf("%d%d",&u,&v);
            edge[u].push_back(v);
            edge[v].push_back(u);
        }
        dfs(1,0);
        printf("Case #%d:\n",++x);
        while (m--) {
            int k;
```

```
                scanf("%d",&k);
                int x,y;
                if (k==1) {
                    int x0,y0;
                    scanf("%d%d%d%d",&x,&y,&x0,&y0);
                    LCT::cut(x,y);
                    LCT::link(x0,y0);
                } else if (k==2) {
                    int val;
                    scanf("%d%d%d",&x,&y,&val);
                    LCT::change(x,y,val);
                } else if (k==3) {
                    int val;
                    scanf("%d%d%d",&x,&y,&val);
                    LCT::add(x,y,val);
                } else if (k==4) {
                    scanf("%d%d",&x,&y);
                    pair<int,int> t=LCT::ask(x,y);
                    if (t.first==-INF) puts("ALL SAME");
                    else printf("%d %d\n",t.first,t.second);
                }
            }
            FOR(i,1,n) edge[i].clear();
        }
}
```

KD 树

```
//线段树套 KD 树
//KD 树,对于子树需要维护区间
//时间复杂度:nsqrt(n)
//最近距离的话,注意剪枝要减得多,用矩形限制
//可以通过对左右估值来确定 query 顺序
//(把 query 的东西放到外面限制)
namespace KDT {
    const double alpha=0.75;
    const int DIM=2;
    struct point {
        int A[DIM],max[DIM],min[DIM];
        int l,r; int size;
        void init() {
            l=r=0; initval();
        }
        void initval() {
            int i; size=1;
            REP(i,DIM) min[i]=max[i]=A[i];
        }
    } T[maxn*30]; int TOT;
    int Cur;
    bool cmp(int x,int y) {
        return T[x].A[Cur]<T[y].A[Cur];
    }
    void update(int x) {
        int i; T[x].initval();
        int l=T[x].l,r=T[x].r;
        if (l) T[x].size+=T[l].size;
        if (r) T[x].size+=T[r].size;
        REP(i,DIM) {
            if (l) {
                T[x].max[i]=max(T[x].max[i],T[l].max[i]);
                T[x].min[i]=min(T[x].min[i],T[l].min[i]);
            }
            if (r) {
                T[x].max[i]=max(T[x].max[i],T[r].max[i]);
                T[x].min[i]=min(T[x].min[i],T[r].min[i]);
            }
        }
    }
    int id[maxn],tot;
    void build(int &x,int l,int r,int cur) { //should have id
        x=0; if (l>r) return;
        int m=(l+r)/2; Cur=cur;
        nth_element(id+l,id+m,id+r+1,cmp);
        x=id[m];
        build(T[x].l,l,m-1,cur^1);
        build(T[x].r,m+1,r,cur^1);
        update(x);
    }
    void getid(int x) { //没有顺序=_=
        id[++tot]=x;
        if (T[x].l) getid(T[x].l);
        if (T[x].r) getid(T[x].r);
    }
    void rebuild(int &x,int cur) {
        tot=0; getid(x);
        build(x,1,tot,cur);
    }
    void insert(int &x,int now,int cur) {
        if (!x) {x=now; return;}
        Cur=cur;
```

```
		if (cmp(now,x)) insert(T[x].l,now,cur^1);
		else insert(T[x].r,now,cur^1);
		update(x);
		if (T[x].size*alpha+3<max(T[T[x].l].size,T[T[x].r].size))
			rebuild(x,cur);
	}
	void addnode(int &x,int px,int py) {
		TOT++; T[TOT].A[0]=px; T[TOT].A[1]=py;
		T[TOT].init(); insert(x,TOT,0);
	}
	int x0,y0,x1,y1;//check 两个=_=
	int check(int x,int y) {
		return x0<=x&&x<=x1&&y0<=y&&y<=y1;
	}
	int ok(point &A) {
		return check(A.A[0],A.A[1]);
	}
	int allin(point &A) {
		return  x0<=A.min[0]&&A.max[0]<=x1&&
			y0<=A.min[1]&&A.max[1]<=y1;
	}
	int allout(point &A) {
		return  A.max[0]<x0||x1<A.min[0]||
			A.max[1]<y0||y1<A.min[1];
	}
	int query(int x) {
		if (!x) return 0;
		if (allin(T[x])) return T[x].size;
		if (allout(T[x])) return 0;
		int ret=0;
		if (ok(T[x])) ret++;
		if (T[x].size==1) return ret;
		ret+=query(T[x].l);
		ret+=query(T[x].r);
		return ret;
	}
}
const int MAX=1e9+7;
struct Tnode {
	int l,r,KD_root;
	Tnode() {l=r=KD_root=0;}
} T[maxn*30]; int cnt;
void update(int &x,int px,int py,int pos,int L,int R) {
	if (!x) x=++cnt;
	KDT::addnode(T[x].KD_root,px,py);
	if (L==R) return;
	int mid=(L+R)/2;
	if (pos<=mid) update(T[x].l,px,py,pos,L,mid);
	else update(T[x].r,px,py,pos,mid+1,R);
}
int query(int x,int k,int L,int R) {
	if (!x) return 0;
	if (L==R) return L;
	int mid=(L+R)/2;
	if (T[x].r) {
		int rk=KDT::query(T[T[x].r].KD_root);
		if (rk<k) return query(T[x].l,k-rk,L,mid);
		return query(T[x].r,k,mid+1,R);
	} return query(T[x].l,k,L,mid);
}
char buffer[36000000],*buf=buffer;
void read(int &x) {
	for (x=0; *buf<48; ++buf);
	while (*buf>=48)x=x*10+*buf-48,++buf;
}
int n,q;
int i,j,k;
int root,lastans;
int main() {
	fread(buffer,1,36000000,stdin);
	read(n); read(q); KDT::TOT=0;
	FOR(i,1,q) {
		int op;
		read(op);
		if (op==1) {
			int x,y,v;
			read(x); read(y); read(v);
			x^=lastans; y^=lastans; v^=lastans;
			update(root,x,y,v,0,MAX);
		} else {
			int x1,y1,x2,y2,k;
			read(x1); read(y1); read(x2); read(y2); read(k);
			x1^=lastans; y1^=lastans;
			x2^=lastans; y2^=lastans;
			k^=lastans;
			KDT::x0=x1; KDT::y0=y1;
			KDT::x1=x2; KDT::y1=y2;
			lastans=query(root,k,0,MAX);
			if (!lastans) puts("NAIVE!ORZzyz.");
			else printf("%d\n",lastans);
```

```
        }
    }
}
```

莫队

sort 时可以按照&1 左往右 or 右往左

树上莫队(套分块)

```
//http://codeforces.com/gym/100962/attachments
//题意是求路径上最小没出现数字
//主要思路是分类,每个点进出各算一次可以消除影响
//点的直接加个 lca 即可
const int SIZE=500;
vector<pair<int,int> > edge[maxn];
int cl[maxn],cr[maxn],val[maxn],dfn[maxn<<1];
int tot;
int dfs(int x,int fa) {
    cl[x]=++tot; dfn[tot]=x;
    for (auto now:edge[x]) if (now.first!=fa) {
            dfs(now.first,x);
            val[now.first]=now.second;
        } cr[x]=++tot; dfn[tot]=x;
}
int block[maxn<<1];
struct node {
    int l,r,id;
} Q[maxn];
int cmp(node a,node b) {
    if (block[a.l]==block[b.l]) return a.r<b.r;
    return block[a.l]<block[b.l];
}
bool vis[maxn];
int cnt[maxn],cur[maxn];//block,now
void change(int x) {
    x=dfn[x]; vis[x]^=1;
    if (vis[x]) {
        if (!cur[val[x]]) cnt[block[val[x]]]++;
        cur[val[x]]++;
    } else {
        cur[val[x]]--;
        if (!cur[val[x]]) cnt[block[val[x]]]--;
    }
}
int ans[maxn];
int L,R;
int main() {
    int n,q;
    int i;
    scanf("%d%d",&n,&q);
    FOR(i,0,n*2+1) block[i]=i/SIZE;
    REP(i,n-1) {
        int u,v,len;
        scanf("%d%d%d",&u,&v,&len); len=min(len,n+1);
        edge[u].push_back(make_pair(v,len));
        edge[v].push_back(make_pair(u,len));
    }
    val[1]=n+1; dfs(1,0);
    REP(i,q) {
        int a,b;
        scanf("%d%d",&a,&b);
        if (cl[a]>cl[b]) swap(a,b);
        if (cr[a]>cr[b]) Q[i].l=cl[a]+1,Q[i].r=cl[b];
        else Q[i].l=cr[a],Q[i].r=cl[b];
        Q[i].id=i;
    }
    sort(Q,Q+q,cmp);
    L=1; R=0;
    REP(i,q) {
        while (L<Q[i].l) {change(L); L++;}
        while (R>Q[i].r) {change(R); R--;}
        while (L>Q[i].l) {L--; change(L);}
        while (R<Q[i].r) {R++; change(R);}
        int now=0;
        while (cnt[now]==SIZE) now++;
        now*=SIZE;
        while (cur[now]) now++;
        ans[Q[i].id]=now;
    }
    REP(i,q) printf("%d\n",ans[i]);
}
```

回滚莫队套分块

```
//北京区域赛,题意是 l->r 的所有内部边的并查集啥的
//回滚分块(然而我没回滚,记录了一下)
//queries 按照左端点排序(有边的要按照我这种方式来排,否则菊花图会卡死)
//cmpu 的时候 v 要倒着,因为要让块外的不受左边影响(same_l and
small_r)
    //然后对于左端点在 block 内部的所有 query,按右端点往右走,走到头
即可
    //这个做法就是按照左分块,然后把右边有效的加进去,再把左边的加
进去就行了
    int SIZE;
    struct node {
        int u,v,id,o;
        node() {};
        node(int _u,int _v,int _id=0):u(_u),v(_v),id(_id) {};
    } to[maxn],re[maxn],queries[maxn];
    int BID[maxn],L[maxn];
    bool cmpu(node A,node B) {
        if (A.u!=B.u) return A.u<B.u;
        //区间为了避免漏掉 r 小的
        if (A.v!=B.v) return A.v>B.v;
        return A.id>B.id;
    }
    bool cmpv(node A,node B) {
        if (A.v!=B.v) return A.v<B.v;
        if (A.u!=B.u) return A.u<B.u;
        return A.id<B.id;
    }
    bool cmpQ(node A,node B) {
        if (A.o!=B.o) return A.o<B.o;
        if (A.v!=B.v) return A.v<B.v;
        if (A.u!=B.u) return A.u<B.u;
        return A.id<B.id;
    }
    int fa[maxn],size[maxn];
    LL Ans[maxn];
    inline int getfa(int x) {
        if (fa[x]==x) return x;
        return fa[x]=getfa(fa[x]);
    }
    int FA[maxn],SZ[maxn],PID[maxn];
    inline int getFA(int x) {
        if (FA[x]==x) return x;
        return FA[x]=getFA(FA[x]);
    }
    inline void update(int u,int pid) {
        if (PID[u]!=pid) {
            int f=getfa(u);
            if (PID[f]!=pid) {
                FA[f]=f;
                PID[f]=pid;
                SZ[f]=size[f];
            } PID[u]=pid; FA[u]=f;
        }
    } int tot=0;
    LL now;
    int main() {
        int T;
        scanf("%d",&T);
        while (T--) {
            int n,m,q,i,j,k;
            scanf("%d%d%d",&n,&m,&q);
            if (q==0) SIZE=m; else SIZE=m/sqrt(q)*2;
            if (!SIZE) SIZE++;
            FOR(i,0,(m+1)/SIZE) L[i]=0;
            FOR(i,1,m+1) {BID[i]=i/SIZE; if (!L[i/SIZE])
L[i/SIZE]=i;}
            FOR(i,1,m) {
                int u,v;
                scanf("%d%d",&u,&v);
                if (u>v) swap(u,v);
                to[i]=node(u,v);
                re[i]=node(u,v);
            } sort(to+1,to+m+1,cmpv);
            sort(re+1,re+m+1,cmpu);
            FOR(i,1,m) {

to[i].o=BID[lower_bound(re+1,re+1+m,to[i],cmpu)-re];
                re[i].o=BID[i];
            }
            FOR(i,1,q) {
                int u,v;
                scanf("%d%d",&u,&v);
                if (u>v) swap(u,v);
                queries[i]=node(u,v,i);

queries[i].o=BID[lower_bound(re+1,re+1+m,queries[i],cmpu)-re];
            } sort(queries+1,queries+q+1,cmpQ);
            FOR(i,1,q) {
                if (i==1||queries[i].o!=queries[i-1].o) { //initialize
                    FOR(j,1,n) fa[j]=j,size[j]=1;
                    j=1; now=0;
                }
                for (; j<=m&&to[j].v<=queries[i].v; j++) {
                    if (to[j].o>queries[i].o) {//sorted by l
                    node &e=to[j];
                    int x=getfa(e.u),y=getfa(e.v);
                    if (x==y) continue; fa[x]=y;
                    now+=(LL)size[x]*size[y];
                    size[y]+=size[x];
                }
            }
            LL ans=now; tot++;
            for (k=L[queries[i].o];
k<=m&&BID[k]==queries[i].o; k++) {
                if (queries[i].u<=re[k].u&&re[k].v<=queries[i].v)
{
                    node &e=re[k];
                    update(e.u,tot); update(e.v,tot);
                    int x=getFA(e.u),y=getFA(e.v);
                    if (x==y) continue; FA[x]=y;
                    ans+=(LL)SZ[x]*SZ[y];
                    SZ[y]+=SZ[x];
                }
            }
            Ans[queries[i].id]=ans;
        }
        FOR(i,1,q) printf("%lld\n",Ans[i]);
    }
}
```

带修改莫队

```
//change 常数大时 size 可以增大
//sort 时先 block,改变顺序可以降低常数
//n^2/3,注意常数
//注意 change 时间时排的顺序
const int SIZE=2500;
struct queries{
    int l,r,t;//pre
    queries(){};
    queries(int _l,int _r,int _t):l(_l),r(_r),t(_t){};
}Q[maxn],S[maxn];
int n,m,q;
int i,j,k;
int a[maxn];
int BLOCK[maxn];
bool cmp(queries &A,queries &B){
    if (BLOCK[A.l]!=BLOCK[B.l]) return
BLOCK[A.l]<BLOCK[B.l];
    if (BLOCK[A.r]!=BLOCK[B.r]) return
BLOCK[A.r]<BLOCK[B.r];
    return (A.t<B.t)^((BLOCK[A.l]^BLOCK[A.r])&1);
}vector<int> V;
inline int getid(int x){return
lower_bound(V.begin(),V.end(),x)-V.begin()+1;}
int L,R,T;
int num[maxn],cnt[maxn];
inline void add(int pos){
    int &T=num[a[pos]];
    cnt[T]--;T++;cnt[T]++;
}inline void del(int pos){
    int &T=num[a[pos]];
    cnt[T]--;T--;cnt[T]++;
}inline void change(int pos,int val){
    if (L<=pos&&pos<=R){del(pos),a[pos]=val,add(pos);}
    else a[pos]=val;
}
int ans[maxn];
int main(){
    scanf("%d%d",&n,&q);
    FOR(i,1,n) scanf("%d",&a[i]),V.push_back(a[i]);
    FOR(i,1,q){
        int op,l,r;
        scanf("%d%d%d",&op,&l,&r);
        if (op==1){
            Q[i]=queries(l,r,i);
        }if (op==2) {
            S[i]=queries(l,r,a[l]);a[l]=r;
            V.push_back(a[l]);
        }
    }sort(V.begin(),V.end());
    V.erase(unique(V.begin(),V.end()),V.end());
    FOR(i,1,n) a[i]=getid(a[i]);
    FOR(i,1,q) if (S[i].t) S[i].r=getid(S[i].r),S[i].t=getid(S[i].t);
    FOR(i,1,max(n,q)) BLOCK[i]=i/SIZE;
    sort(Q+1,Q+q+1,cmp);
    L=1;R=0;T=q;cnt[0]=INF;
    FOR(i,1,q) if (Q[i].t){
        while (T<Q[i].t){T++;if (S[T].t) change(S[T].l,S[T].r);}
        while (T>Q[i].t){if (S[T].t) change(S[T].l,S[T].t);T--;}
        while (L<Q[i].l){del(L);L++;}
        while (R>Q[i].r){del(R);R--;}
        while (L>Q[i].l){L--;add(L);}
        while (R<Q[i].r){R++;add(R);}
```

```
            int now=0;
            while (cnt[now]) now++;
            ans[Q[i].t]=now;
        }FOR(i,1,q) if (ans[i]) printf("%d\n",ans[i]);
    }
```

二次离线莫队

```
    // 题意: 区间 A[i]%B[i]=0 的对数
    // 做法: 第十四分块, 二次离线莫队然后再离线 bigsmall 算贡献, 用 fractional cascading(分散层叠也行)
    // 分散层叠: 每一层保存这层所有信息+一半下一层的信息(position)
    // 那么只 lowerbound, 然后 while 回去就行了, 复杂度 k(层数)+log(lowerbound)
    // 还是得老老实实算, 不能按贡献算, 因为有可能向前和向后的贡献不一致(?)
    // [l,r] to [l,r'] = [1,  r(x)] to [(x+1),r'] - [1,l-1] to [r+1,r'] = sum_l[r']-sum_l[r]-[1,l-1] to [r+1,r']
    // [l,r] to [l',r] =-[l(x+1),n] to [(x),l'-1] + [r+1,n] to [l,l'-1] = sum_r[l']-sum_r[l]+[r+1,n] to [l,l'-1](from r+1,rev)
    // 下面式子是不对称的(贡献一致可以减常数)
    // [l,r] to [l',r] = [1,   (x)] to [(x),l'-1] - [1,  r] to [l,l'-1] = sumlx[l'-1]-sumlx[l-1]-[1,r] to [l,l'-1]
    // 二次离线的作用是如果可以 O(1)查询某个点的值, O(sqrt)更新, 就可以 for 一遍范围直接加起来!
    // 二次离线之后还是要再离线算其他的贡献
    struct node {
        int l,r,pos,type;
    } Q[maxn];
    vector<node> Ql[maxn],Qr[maxn];
    int A[maxn];
    int BLOCK[maxn];// 莫队 sqrt
    const int SIZE=300;
    int sumx[maxn],sumy[maxn];
    ll suml[maxn],sumr[maxn];
    ll base[maxn],ans[maxn];//more
    int C[maxn];//front
    int SIZE_B=50;// big_small, 这个常数也太大了
    vector<int> fac[maxn];
    int main() {
        int i;
        int n,q;
        scanf("%d%d",&n,&q);
        FOR(i,1,n) scanf("%d",&A[i]);
        FOR(i,1,n) BLOCK[i]=i/SIZE;
        FOR(i,1,q) scanf("%d%d",&Q[i].l,&Q[i].r),Q[i].pos=i;
        sort(Q+1,Q+1+q,[&](node &x,node &y) {
            if (BLOCK[x.l]!=BLOCK[y.l]) return BLOCK[x.l]<BLOCK[y.l];
            else return bool((x.r<y.r)^(BLOCK[x.l]&1));
        });
        int l=1,r=0;
        FOR(i,1,q) {
            int l_l=Q[i].l,r_r=Q[i].r,base;
            base=-1; if (r>r_r) swap(r,r_r),base*=-1;
            if (r<r_r) Ql[l-1].push_back(node{r+1,r_r,i,base}); r=Q[i].r;
            base=1; if (l>l_l) swap(l,l_l),base*=-1;
            if (l<l_l) Qr[r+1].push_back(node{l,l_l-1,i,base}); l=Q[i].l;
        }
        int k;
        FOR(i,1,100000) {
            for (int k=i; k<=100000; k+=i) fac[k].push_back(i);
        } FOR(i,1,100000) C[i]=0;
        //first
        FOR(i,1,n) {//x=ky or {kx=y and x>sqrt}
            suml[i]=C[A[i]];
            for (int k:fac[A[i]]) C[k]++;
            if (A[i]>SIZE_B)
                for (int k=A[i]; k<=100000; k+=A[i]) C[k]++;
            for (auto now:Ql[i])
                FOR(k,now.l,now.r) base[now.pos]+=C[A[k]]*now.type;
        } FOR(i,1,100000) C[i]=0;
        rFOR(i,1,n) {
            sumr[i]=C[A[i]];
            for (int k:fac[A[i]]) C[k]++;
            if (A[i]>SIZE_B)
                for (int k=A[i]; k<=100000; k+=A[i]) C[k]++;
            for (auto now:Qr[i])
                FOR(k,now.l,now.r) base[now.pos]+=C[A[k]]*now.type;
        }
        //second, BLOCK
        FOR(k,1,SIZE_B) {
            FOR(i,1,n) {//注意 r 可能要另算,这个题对称所以 ok
                sumx[i]=sumx[i-1]+(A[i]==k);
                sumy[i]=sumy[i-1]+(A[i]%k==0);
            }
            FOR(i,1,n) {
                if (A[i]%k==0)
suml[i]+=sumx[i-1],sumr[i]+=sumx[n]-sumx[i];
                for (auto now:Ql[i])

base[now.pos]+=(ll)sumx[i]*(sumy[now.r]-sumy[now.l-1])*now.t
ype;
                for (auto now:Qr[i])

base[now.pos]+=(ll)(sumx[n]-sumx[i-1])*(sumy[now.r]-sumy[no
w.l-1])*now.type;
            }
        }
        FOR(i,1,n) suml[i]+=suml[i-1];
        rFOR(i,1,n) sumr[i]+=sumr[i+1];
        l=1,r=0;
        FOR(i,1,q) {
            int l_l=Q[i].l,r_r=Q[i].r,x=Q[i].pos;

base[i]=suml[r_r]-suml[r]+sumr[l_l]-sumr[l]+base[i]+base[i-1];
            ans[x]=base[i]+r_r-l_l+1;
            l=l_l; r=r_r;
        }
        FOR(i,1,q) printf("%lld\n",ans[i]);
    }
```

用 set 维护凸包

```
    /* 这是抄的维护上半凸壳 */
    // 最大值,query 的 k 要求>0
    bool Q;
    struct Line {
        mutable LL a,b,k;
        bool operator<(const Line &o)const {
            return Q?k<o.k:a<o.a;
        }
    };
    struct convexHull:public multiset<Line> {
        LL div(LL a,LL b) {
            return a/b-((a^b)<0&&a%b);
        }
        bool getK(iterator x,iterator y) {
            if (y==end()) {x->k=INFF; return 0;}
            if (x->a==y->a) x->k=x->b>y->b?INFF:-INFF;
            else x->k=div(y->b-x->b,x->a-y->a);
            return x->k>=y->k;
        }
        void insPos(LL a,LL b) {
            auto z=insert({a,b,0}); auto y=z++,x=y;
            while (getK(y,z)) z=erase(z);
            if (y!=begin()&&getK(--x,y)) getK(x,erase(y));
            while ((y=x)!=begin()&&(--x)->k>=y->k)
                getK(x,erase(y));
        }
        LL query(LL x) {
            assert(size());
            Q=1; auto now=lower_bound({0,0,x}); Q=0;
            return now->a*x+now->b;
        }
    };
```

李超树

```
    //李超树最主要的作用在于维护线段,而不是直线!
    //维护 l<=x<=r 时下放线段,时间复杂度两个 log!
    //这里是最大值
    double cross(double k1,double b1,double k2,double b2) {
        if (abs(k1-k2)<eps) return INF;
        return (b2-b1)/(k1-k2);
    }
    int flag[maxn*4];
    double tagk[maxn*4],tagb[maxn*4];
    void ins(int x,double k,double b,int l,int r,int id,int L,int R) {
        if (l<=L&&R<=r) {
            if (!flag[x]) tagk[x]=k,tagb[x]=b,flag[x]=id;
            else {
                int mid=(L+R)/2;
                double ini_l=tagk[x]*L+tagb[x],now_l=k*L+b;
                double ini_r=tagk[x]*R+tagb[x],now_r=k*R+b;
                if (ini_l>=now_l&&ini_r>=now_r) return;
                if (ini_l<=now_l&&ini_r<=now_r)
tagk[x]=k,tagb[x]=b,flag[x]=id;
                else {
                    double pos=cross(k,b,tagk[x],tagb[x]);//交点 x
坐标
                    if
((pos<=mid&&ini_l>=now_l)||(pos>mid&&ini_r>=now_r)) { //坐标
低的下放,平的直接留下就行
                        swap(tagk[x],k);
                        swap(tagb[x],b);
                swap(flag[x],id);
            } if (pos<=mid) ins(x<<1,k,b,l,r,id,L,mid);
            else ins(x<<1|1,k,b,l,r,id,mid+1,R);
        }
    }
  } else {
    int mid=(L+R)/2;
    if (l<=mid) ins(x<<1,k,b,l,r,id,L,mid);
    if (mid<r) ins(x<<1|1,k,b,l,r,id,mid+1,R);
  }
}
double ans; int id;
void que(int x,int pos,int L,int R) {
  if (flag[x]) {
    double now=tagk[x]*pos+tagb[x];
    if (now-ans>eps||(now-ans>-eps&&id>flag[x])) {
      ans=now,id=flag[x];
    }
  }
  if (L==R) return;
  int mid=(L+R)/2;
  if (pos<=mid) que(x<<1,pos,L,mid);
  else que(x<<1|1,pos,mid+1,R);
}
```

线性基(套路)

```
namespace LB {
  typedef long long BaseType;
  const int MaxBit=63;
  struct L_B {
    BaseType b[MaxBit]; bool have_0;
    L_B() {clear();}
    void clear() {memset(b,0,sizeof(b)); have_0=0;}
    BaseType XORMIN(BaseType x) {
      int i;
      rREP(i,MaxBit) if ((b[i]^x)<x) x^=b[i];
      return x;
    }
    BaseType XORMAX(BaseType x) {
      int i;
      rREP(i,MaxBit) if ((b[i]^x)>x) x^=b[i];
      return x;
    }
    void insert(BaseType x) {
      int i;
      if (!have_0&&!XORMIN(x)) have_0=1;
      rREP(i,MaxBit) if ((x>>i)&1) {
        if (!b[i]) b[i]=x; x^=b[i];
      }
    }
    void rebuild() {
      int i,j;
      rREP(i,MaxBit) rREP(j,i) if ((b[i]>>j)&1)
b[i]^=b[j];
    }
    BaseType querykth(BaseType k) {
      BaseType ret=0; int i; k-=have_0;
      REP(i,MaxBit) if (b[i]) {if (k&1) ret^=b[i]; k>>=1;}
      if (k) return -1;
      return ret;
    }
  } A;
  //求交 merge 的思路: 只要 A 中 merge 之后的线性无关组
  L_B merge(const L_B &A,const L_B &B) {
    int i,j; L_B ret; ret.clear();
    static BaseType
base[MaxBit],tmp[MaxBit];//previous_A
    REP(i,MaxBit) tmp[i]=A.b[i],base[i]=1ll<<i;
    REP(i,MaxBit) if (B.b[i]) {//正着反着应该没区别
      BaseType now=B.b[i];
      bool okay=1; BaseType k=0;//base; A
      rREP(j,MaxBit) if ((now>>j)&1) {
        if (tmp[j]) {
          now^=tmp[j]; k^=base[j];
        } else {
          tmp[j]=now; base[j]=k; okay=0; break;
        }
      }
      if (okay) {
        BaseType should=0;
        REP(j,MaxBit) if ((k>>j)&1) should^=A.b[j];
        ret.insert(should);
      }
    }
    return ret;
  }
}
```

手写 BITSET

```
struct BITSET {
    vector<ULL> V;
    void set(int x,int k) {
        assert((int)V.size()>x/64);
        if (k) V[x/64]|=1ull<<(x&63);
        else V[x/64]&=~(1ull<<(x&63));
    }
    void resize(int x) {
        V.resize((x-1)/64+1,0);
    }
    int get(int x) {
        return (V[x/64]>>(x&63))&1;
    }
    bool operator < (const BITSET &B) const {
        int i;
        REP(i,(int)V.size()) if (V[i]!=B.V[i]) return V[i]<B.V[i];
        return 0;
    }
    BITSET const doit(int size,int F[65536]) const {//相邻两
位合并
        BITSET ret; int i;
        ret.resize(size/2);
        REP(i,(int)V.size()) {
            if (i&1) {
                ret.V[i/2]|=((ULL)F[V[i]&65535]<<32)

|((ULL)F[(V[i]>>16)&65535]<<40)

|((ULL)F[(V[i]>>32)&65535]<<48)
                                    |((ULL)F[(V[i]>>48)]<<56);
            } else {
                ret.V[i/2]|=((ULL)F[V[i]&65535])

|((ULL)F[(V[i]>>16)&65535]<<8)

|((ULL)F[(V[i]>>32)&65535]<<16)
                                    |((ULL)F[(V[i]>>48)]<<24);
            }
        } return ret;
    }
    void print() {
        int i;
        REP(i,(int)V.size()) pr2(V[i],64);
    }
};
```

杨表

```
//题意: 选 5 个 subsquence 和最大
//杨表: 单调**子序列个数最大多少个
//杨表做法: 直接替换原数列中比这个大的位置,
//      然后直接将多出来的往下放即可
//正确性: 可以将这个点后面连的所有东西放下边,
//      相当于连个边,相当于最优选择
ll ans;
map<int,ll> MP[5];//pos,cnts
void update(int x,int y,int dep) {//x,cnt
    if (dep==5) return;
    while (y) {
        map<int,ll>::iterator it=MP[dep].upper_bound(x);
        if (it==MP[dep].end()) {
            ans+=y;
            MP[dep][x]+=y; break;
        } pair<int,int> now=*it;
        MP[dep].erase(it);
        ll down=min(now.second,y);
        y-=down; now.second-=down;
        if (now.second) MP[dep][now.first]+=now.second;
        MP[dep][x]+=down;
        update(now.first,down,dep+1);
    }
}
int main(){
    int T,_; T=1;
    scanf("%d",&T);
    FOR(_,1,T){
        int i,n;
        scanf("%d",&n); ans=0;
        REP(i,5) MP[i].clear();
        FOR(i,1,n) {
            int k; scanf("%d",&k);
            update(k,k,0);
            printf("%lld%c",ans," \n"[i==n]);
        }
    }
}
```

图论

二分图匹配

//最小不相交路径覆盖<=>节点数-拆点以后二分图最大匹配

//最小相交路径覆盖<=>所有能走到的节点连边，然后节点数-拆点以后匹配

```
vector<int>edge[N];
int used[N];
```

注意数组的标号，必须满足二分图的条件

```
int matching[N];
bool dfs(int u){
    int v,i;
    REP(i,edge[u].size()){
        v=edge[u][i];
        if (!used[v]){
            used[v]=1;
            if (matching[v]==-1||dfs(matching[v])){
                matching[v]=u;
                matching[u]=v;
                return 1;
            }
        }
    }
    return 0;
}
int DFS(){
    int ans=0;
    memset(matching,-1,sizeof(matching));
    int u;
    FOR(u,1,n){
        if (matching[u]==-1){
            memset(used,0,sizeof(used));
            if (dfs(u)) ans++;
        }
    }
    return ans;
}
```

注意数组的标号，必须满足二分图的条件

```
queue<int> Q;
int prev[N];//两格
int matching[N];//结果
int check[N];//matchright
int BFS(){
    int ans=0;
    memset(matching,-1,sizeof(matching));
    memset(check,-1,sizeof(check));
    FOR(i,1,n){
        if (matching[i]==-1){
            while (!Q.empty()) Q.pop();
            Q.push(i);
            prev[i]=-1;
            bool flag=false;
            while (!Q.empty()&&!flag){
                int u=Q.front();Q.pop();
                for (j=0;!flag&&j<edge[u].size();j++){
                    int v=edge[u][j];
                    if (check[v]!=i){
                        check[v]=i;
                        Q.push(matching[v]);
                        if (matching[v]!=-1)
prev[matching[v]]=u;
                        else{
                            flag=1;
                            int d=u,e=v;
                            while (d!=-1){
                                int t=matching[d];
                                matching[d]=e;
                                matching[e]=d;
                                d=prev[d];
                                e=t;
                            }
                        }
                    }
                }
            }
            if (matching[i]!=-1) ans++;
        }
        }
        return ans;
    }
```

Hall 定理

// 题意: N 个人,M 个椅子,每个人只能坐[1,Li]|[Ri,M],求最多能坐多少人

// hall 定理: 二分图; A->B (A<B)完美匹配当且仅当 A 中每 k 个在 B 中连着有至少 k 个点

// 引理(不常用): 如果 A 中每个连着最少 t 条边, B 中每个连着最多 t 条边，那么存在完美匹配; t 任意

// 对于这个题来说: 最终选择的座位比人少; 任意座位集合 A; B: [1,Lx][Rx,M]

// 座位当作 A，用定理，所有区间满足: 对人的集合 B, A->B ,|A|>=$加完边$的 B 求下|A|-|B|>=0

// 枚举 A 的端点，求: B 的 size 最大值即可!

```
    int MIN[maxn],lazy[maxn];
    inline void add(int x,int val){
        lazy[x]+=val;MIN[x]+=val;
    }void update(int x,int l,int r,int val,int L,int R){
        if (l<=L&&R<=r){add(x,val);return;}
        if (lazy[x]){
            add(x<<1,lazy[x]);
            add(x<<1|1,lazy[x]);
            lazy[x]=0;
        }int mid=(L+R)/2;
        if (l<=mid) update(x<<1,l,r,val,L,mid);
        if (mid<r) update(x<<1|1,l,r,val,mid+1,R);
        MIN[x]=min(MIN[x<<1],MIN[x<<1|1]);
    }int n,m;
    vector<int> have[maxn];
    int i,j,k;
    int l,r;
    int ans;
    int main(){
        scanf("%d%d",&n,&m);
        FOR(i,1,n){
            scanf("%d%d",&l,&r);
            have[l].push_back(r);
        }
        FOR(i,1,m) update(1,i,i,m-i+1,1,m+1);
        ans=min(0,m-n);//为啥会有这个问题呢
        FOR(i,0,m){
            if (i!=0) update(1,i+1,m+1,1,1,m+1);
            for (int r:have[i])
                update(1,i+1,r,-1,1,m+1);
            ans=min(ans,MIN[1]);
        }printf("%d\n",-ans);
    }
```

KM 二分图最大权匹配

```
ll g[maxn][maxn];
ll lx[maxn],ly[maxn],slack[maxn];
int linky[maxn],par[maxn];
bool visy[maxn];
void augment(int root){
    std::fill(visy+1,visy+n+1,false);
    std::fill(slack+1,slack+n+1,INFF);
    int py; linky[py=0]=root;
    do{
        visy[py]=true;
        int x=linky[py],_y=0,y; ll d=INFF;
        FOR(y,1,n) if (!visy[y]){
            int tmp=lx[x]+ly[y]-g[x][y];
            if (tmp<slack[y]){
                slack[y]=tmp; par[y]=py;
            } if (slack[y]<d) {
                d=slack[y]; _y=y;
            }
        } FOR(y,0,n){
            if (visy[y]){
                lx[linky[y]]-=d;
                ly[y]+=d;
            } else slack[y]-=d;
        } py=_y;
    } while (linky[py]!=-1);
    do {
        int pre=par[py];
        linky[py]=linky[pre];
        py=pre;
    } while (py);
}
ll KM() {
    int i,y;
    FOR(i,1,n) {
        lx[i]=0; ly[i]=0; linky[i]=-1;
        FOR(y,1,n) max_(lx[i],g[i][y]);
    } ll ret=0;
    FOR(i,1,n) augment(i);
    FOR(i,1,n) ret+=g[linky[i]][i];
    return ret;
}
int main() {
    int T,_T;
    scanf("%d",&T);
    FOR(_T,1,T) {
        scanf("%d",&n);
        int i,j;
        FOR(i,1,n) FOR(j,1,n) {
            int x;
            scanf("%d",&x);
            g[i][j]=-x;
        } ll ans=-KM();
        // printf("%d\n",ans);
        printf("Case #%d: %I64d\n",_T,ans);
    }
}
```

最短路

Dijkstra：略

SPFA DFS(只用于判负环)

```
struct node{
    int n,d;
    node(){}
    node(int a,int b):n(a),d(b){}
    bool operator<(const node&a)const{
        if (d==a.d) return n<a.n;
        return d>a.d;
    }
};
vector<node> edge[maxn];
int dis[maxn],n,m;
bool vis[maxn];
bool spfa(int u){
    int i;
    vis[u]=1;
    REP(i,edge[u].size()){
        node v=edge[u][i];
        if (dis[u]+v.d<dis[v.n]){
            dis[v.n]=dis[u]+v.d;
            if (vis[v.n]) return 1;
            else {
                dis[v.n]=dis[u]+v.d;
                if (spfa(v.n)) return 1;
            }
        }
    }
    vis[u]=0;
    return 0;//judge negative ring
}
int s,t;
int u,v,len;
int main(){
    int i,j,k;
    while (~scanf("%d%d",&n,&m)){
        FOR(i,1,n) edge[i].clear();
        REP(i,m){
            scanf("%d%d%d",&u,&v,&len);
            edge[u].push_back(node(v,len));
            edge[v].push_back(node(u,len));
        }
        FOR(i,1,n) dis[i]=INF;dis[1]=0;
        FOR(i,1,n) vis[i]=0;
        spfa(1);
        FOR(i,2,n) printf("%d ",dis[i]==INF?-1:dis[i]);
        puts("");
    }
}
```

差分约束系统

//主要在于建图

//连边 u->v,len <==> val(v)-val(u)<=len

//其他的都要化成这种形式 int n,m;

//最好 spfa!(可能负环)

01 分数规划

//2017-harbin-K

//选出 k 个区间，使得这 k 个区间全覆盖，而且 sigmaA/sigmaB 最小

//俩 log dp TLE

//做法：建最短路，01 分数规划玄学过题

```
struct node {
    int n;
    double d;
```

```
        node() {}
        node(int _n,double _d):n(_n),d(_d) {};
        bool operator<(const node &A)const {
            if (d==A.d) return n<A.n;
            return d>A.d;
        }
    };
    struct node_e {
        int n,A,B;
        double d;
        node_e(int _n,int _A,int _B,double
_d):n(_n),A(_A),B(_B),d(_d) {}
    };
    vector<node_e> edge[maxn];
    int dis[maxn];
    int preA[maxn],preB[maxn];
    void dij(int s,int n) {
        int i;
        FOR(i,1,n) dis[i]=INF;
        dis[s]=0;
        priority_queue<node> Q;
        Q.push(node(s,dis[s]));
        while (Q.size()) {
            node x=Q.top();
            Q.pop();
            for (auto &y:edge[x.n]) {
                if (dis[y.n]>x.d+y.d) {
                    dis[y.n]=x.d+y.d;
                    Q.push(node(y.n,dis[y.n]));
                    preA[y.n]=preA[x.n]+y.A;
                    preB[y.n]=preB[x.n]+y.B;
                }
            }
        }
    }
    int n,t;
    int S[maxn],T[maxn],A[maxn],B[maxn];
    double check(double x) {
        int i;
        double allA=0,allB=0;
        FOR(i,1,t+1)
        edge[i].clear();
        FOR(i,1,n) {
            if (A[i]-B[i]*x<=0) {
                allA+=A[i];
                allB+=B[i];
                edge[S[i]].emplace_back(node_e(T[i]+1,0,0,0));
            } else
edge[S[i]].emplace_back(node_e(T[i]+1,A[i],B[i],A[i]-B[i]*x));
        }
        FOR(i,1,t)
        edge[i+1].emplace_back(node_e(i,0,0,0));
        dij(1,t+1);
        allA+=preA[t+1];
        allB+=preB[t+1];
        return allA/allB;
    }
    int main() {
        int i,j,m,x,_T;
        scanf("%d",&_T);
        while (_T--) {
            scanf("%d%d",&n,&t);
            FOR(i,1,n)
            scanf("%d%d%d%d",&S[i],&T[i],&A[i],&B[i]);
            double ans=100;
            while (1) {
                double now=check(ans);
                if (abs(now-ans)<0.001) break;
                ans=now;
            }
            printf("%.3lf\n",ans);
        }
        return 0;
    }
```

切比雪夫(曼哈顿)距离最小生成树

//最小曼哈顿距离生成树
//按照 45 度 4 个方向排序，最近的两个点连边即可
//最大曼哈顿距离生成树是维护最远的点的距离（四个方向的）
//Kruskal(有道分治题用的 Boruvka，和这个思想也类似)
//注意理解并查集的内涵，每次找最短的路也可以通过其他方式来找到

切比雪夫距离转曼哈顿距离：
切比雪夫距离：max(|x1-x2|,|y1-y2|);
曼哈顿距离：|x1-x2|+|y1-y2|
转化方式：旋转 45 度然后/2
(x,y)->((x+y)/2,(x-y)/2)

曼哈顿距离最小生成树：

按照 45 度 4 个方向排序，最近的两个点连边即可

swap 方向代码：

```
int a[MAXN],b[MAXN];
tot = 0;
for (int dir = 0; dir < 4; dir++) {
//4 种坐标变换
    if (dir == 1 || dir == 3) {
        for (int i = 0; i < n; i++) swap(p[i].x,p[i].y);
    } else if (dir == 2) {
        for (int i = 0; i < n; i++) p[i].x = -p[i].x;
    }
    sort(p,p+n,cmp);
    for (int i = 0; i < n; i++)
        a[i] = b[i] = p[i].y - p[i].x;
    sort(b,b+n);
    int m = unique(b,b+n) - b;
    for (int i = 1; i <= m; i++) bit[i].init();
    for (int i = n-1 ; i >= 0; i--) {
        int pos = lower_bound(b,b+m,a[i]) - b + 1;
        int ans = ask(pos,m);
        if (ans != -1)
            addedge(p[i].id,p[ans].id,dist(p[i],p[ans]));
        update(pos,p[i].x+p[i].y,i);
    }
}
```

笛卡尔树

2017hdu 多校 1 给定区间:

```
int L[maxn],R[maxn];
pii S[maxn]; int top;
int fa[maxn],id[maxn];//id: topo
bool buildtree(int n){//return 1: wrong!
    static int i,l[maxn],r[maxn],p[maxn],top;
    FOR(i,1,n) id[i]=i; top=0;
    sort(id+1,id+1+n,[](int i,int j){
        if (L[i]!=L[j]) return L[i]<L[j];
        return R[i]>R[j];
    }); top=1;
    l[1]=1; r[1]=n; p[1]=0;
    FOR(i,1,n){
        if (L[id[i]]!=l[top]||r[top]!=R[id[i]]) return 1;
        fa[id[i]]=p[top]; top--;
        if (id[i]<R[id[i]]) {
            ++top,p[top]=id[i];
            l[top]=id[i]+1,r[top]=R[id[i]];
        } if (L[id[i]]<id[i]) {
            ++top,p[top]=id[i];
            l[top]=L[id[i]],r[top]=id[i]-1;
        }
    }
    // FOR(i,1,n) printf("%d ",id[i]);puts("");
    // FOR(i,1,n) printf("%d ",fa[i]);puts("");
    return 0;//okay
}
LL inv[1000002];//inverse
LL fac[1000002];//Factorial
LL C(int n,int m){
    return fac[n]*inv[m]%M*inv[n-m]%M;
}
int sz[maxn],s[maxn];
int main() {
    int _t=0;
    int i;
    fac[0]=1;
    FOR(i,1,1000000) fac[i]=i*fac[i-1]%M;
    inv[0]=inv[1]=1;
    FOR(i,2,1000000) inv[i]=(M-M/i)*inv[M%i]%M;
    FOR(i,1,1000000) inv[i]=inv[i]*inv[i-1]%M;// inv(n!)
    while (1){
        read(n);
        if (Istream::IOerror) break;
        int i;
        FOR(i,1,n) read(L[i]);
        FOR(i,1,n) read(R[i]);
        int ans=1;
        if (buildtree(n)) ans=0;
        FOR(i,1,n) sz[i]=1;
        rFOR(i,1,n) sz[fa[id[i]]]+=sz[id[i]];
        FOR(i,1,n) s[i]=sz[i]-1;
        rFOR(i,2,n) {

mul_(ans,C(s[fa[id[i]]],sz[id[i]])),s[fa[id[i]]]-=sz[id[i]];
        }
        printf("Case #%d: %d\n",++_t,ans);
    }
}
```

2018hdu 多校 1 给定数字:

```
// 按照 A 从大到小建笛卡尔树
int A[maxn],fa[maxn],id[maxn];//id: topo
```

```
void buildtree(int n){
    static int S[maxn],top,tot,i;
    tot=top=0;
    FOR(i,1,n){
        int now=0;
        while (top&&A[S[top]]<A[i]){
            if (now) fa[now]=S[top],id[++tot]=now;//pop
            now=S[top];  top--;
        } S[++top]=i;
        if (now) fa[now]=S[top],id[++tot]=now;//pop
    } int now=0;
    while (top){
        if (now) fa[now]=S[top],id[++tot]=now;
        now=S[top]; top--;
    } fa[now]=0; id[++tot]=now;
    reverse(id+1,id+1+n);// 变成正的
}
int inv[maxn];
int sz[maxn];//求树的 size
int main() {
    int T,_t;
    int i;
    FOR(i,1,1000000) inv[i]=powMM((ll)i,M-2);
    scanf("%d",&T);
    FOR(_t,1,T){
        scanf("%d",&n);
        FOR(i,1,n) scanf("%d",&A[i]);
        buildtree(n);
        int ans=(ll)n*inv[2]%M;
        FOR(i,1,n) sz[i]=1;
        rFOR(i,2,n) sz[fa[id[i]]]+=sz[id[i]];
        FOR(i,1,n) mul_(ans,inv[sz[i]]);
        printf("%d\n",ans);
    }
}
```

强连通分量 tarjan

```
struct Edge {
    int to,next;
    Edge(int _to=0,int _next=-1):to(_to),next(_next) {};
} edge[maxn*2];
int head[maxn],etot;
inline void addedge(int u,int v) {
    edge[++etot]=Edge(v,head[u]);
    head[u]=etot;
}
//lowlink 是说,遇到的 min
//无向图:
//u 割点:low[v]>=dfn[u];(表示能到的点都在之后)
//u-v 割边(桥):low[v]>dfn[u];(要在 u-v 处得到)
//块:low[u]==dfn[u];(最终从 stack 取出 x)
//dfs 时注意 fa 和重边处理
//无向图不用 vis 这个东西=_=,vis 是为了避免横叉边
vector<int> nodes[maxn];
int cnt;
int dfn[maxn],low[maxn],tot;
bool vis[maxn];//instack
int S[maxn],top;
int id[maxn];
void tarjan(int x,int fa) {
    low[x]=dfn[x]=++tot;
    S[++top]=x;
    vis[x]=1;
    for(int i=head[x]; ~i; i=edge[i].next) {
        int v=edge[i].to;
        if(v==fa) continue;
        if(!dfn[v]) {
            tarjan(v,x);
            low[x]=min(low[x],low[v]);
        } else if(vis[v])
            low[x]=min(low[x],dfn[v]);
    }
    if(low[x]==dfn[x]) {
        cnt++;
        while(1) {
            int now=S[top--];
            vis[now]=0;
            id[now]=cnt;
            nodes[cnt].push_back(now);
            if(now==x) break;
        }
    }
}
```

支配树

```
//lowlink 是说,遇到的 min
//无向图:
//u 割点:low[v]>=dfn[u];(表示能到的点都在之后)
```

```
//u-v 割边(桥):low[v]>dfn[u];(要在 u-v 处得到)
//块:low[u]==dfn[u];(最终从 stack 取出 x)
//dfs 时注意 fa 和重边处理
//有向图:
//DAG 上的割边:u-v:cnt[u]*cnt[v]==cnt[t](mod?)
//DAG 上的割边是固定的,也就是说求出来以后最短路是一样长的
//有环割边:将边变成点,然后跑支配树即可
//支配树:(注意,由于可能有到达不了的节点,初始化时注意答案更新)
//半必经点(semi=mindep{通过非树枝边 fa})定理:(semi[x]=id[temp]),
//temp=min(temp,dfn[pre]),dfn[x]>dfn[pre](树枝边|前向边)
//temp=min{temp,dfn[semi[ancestor_pre(fa)]]}
//dfn[x]<dfn[pre](横叉边|后向边)
//必经点(idom)定理:y=id[min{dfn[z]}],z:semi_path 上的点
//idom[x]=semi[x],semi[x]==semi[y]
//idom[x]=idom[y],semi[x]!=semi[y]
struct Edge {
    int to,next;
    Edge(int _to=0,int _next=-1):to(_to),next(_next) {};
} edge[maxn*4];
int head[maxn],pre[maxn],dom[maxn],etot; //edges
inline void addedge(int head[],int u,int v) {
    edge[++etot]=Edge(v,head[u]);
    head[u]=etot;
}
int dfn[maxn],tot,par[maxn]; //dfs-tree
int Fa[maxn],best[maxn]; //disjoint-set
int semi[maxn],id[maxn],idom[maxn]; //dom-tree
inline int getfa(int x) {
    if(Fa[x]==x) return x;
    int F=getfa(Fa[x]);
    if(dfn[semi[best[x]]]>dfn[semi[best[Fa[x]]]])
        best[x]=best[Fa[x]];
    return Fa[x]=F;
}
void dfs(int x) {
    dfn[x]=++tot;
    id[tot]=x;
    for(int i=head[x]; ~i; i=edge[i].next) {
        int v=edge[i].to;
        if(!dfn[v]) par[v]=x,dfs(v);
    }
}
void tarjan(int n) {
    int i;
    FOR(i,1,n) dom[i]=-1;
    FOR(i,1,n) best[i]=semi[i]=Fa[i]=i;
    rFOR(i,2,tot) {
        int x=id[i];
        for(int j=pre[x]; ~j; j=edge[j].next) {
            int v=edge[j].to;
            if(!dfn[v]) continue; //could not reach
            getfa(v); //pre_dfn:not changed
            if(dfn[semi[best[v]]]<dfn[semi[x]])
                semi[x]=semi[best[v]];
        }
        addedge(dom,semi[x],x);
        Fa[x]=par[x];
        x=id[i-1];
        for(int j=dom[x]; ~j; j=edge[j].next) { //path
            int v=edge[j].to;
            getfa(v); //id[min{dfn[z]}];
            if(semi[best[v]]==x) idom[v]=x;
            else idom[v]=best[v];
        }
    }
    FOR(i,2,tot) {
        int x=id[i];
        if(idom[x]!=semi[x]) idom[x]=idom[idom[x]];
    }
}
LL n,m;
LL CNT[maxn];
LL solve() {
    LL ret=(LL)tot*(tot-1)/2;
    int i;
    rFOR(i,2,tot) {
        int x=id[i];
        CNT[x]++;
        if(idom[x]==1) ret-=CNT[x]*(CNT[x]-1)/2;
        else CNT[idom[x]]+=CNT[x];
    }
    return ret;
}
int main() {
    int i;
    scanf("%d%d",&n,&m);
    FOR(i,1,n) head[i]=pre[i]=-1;
    FOR(i,1,n) dfn[i]=id[i]=idom[i]=0;etot=tot=0;
    FOR(i,1,m) {
        int u,v;
        scanf("%d%d",&u,&v);
        addedge(head,u,v);
        addedge(pre,v,u);
    }
    dfs(1);
    tarjan(n);
//    FOR(i,1,n) printf("%2d ",par[i]);puts("");
//    FOR(i,1,n) printf("%2d ",id[i]);puts("");
//    FOR(i,1,n) printf("%2d ",idom[i]);puts("");
    printf("%lld\n",solve());
}
```

边双连通分量 仙人掌图

```
// 2018hdu 多校 5A
// 题意: 每两个点之间只能有两条路径
// 也就是仙人掌
// 求\sum flow(i,j)^i^j, i<j
// 做法: 有环的话, 一定会切掉环上的一个边
// 所以把贡献加到其他里, lca_sum(count)
// 2019nowcoder6I 题意: 仙人掌, 从 1 号点开始选
// 如果存在 u-v 且 u 在某轮选了, v 在下一轮选择概率 w
// 求期望概率 p. 做法是 p=1-p1p2...-pipi+1...
// 然后仙人掌 dp 即可
// 注意一下方向
struct edges {
    int u,v,len;
} e[maxn];
// vector<edges> E;
namespace tarjan { // 边双连通分量, 这里是在做仙人掌
    struct Edge {
        int to,next,id;
        Edge(int _to=0,int _next=-1,int
_id=0):to(_to),next(_next),id(_id) {};
    } edge[maxn*2];
    int head[maxn],etot;
    inline void addedge(int u,int v,int id) {
        edge[++etot]=Edge(v,head[u],id); head[u]=etot;
    }
    int dfn[maxn],low[maxn],tot;
    bool vis[maxn],used[maxn];
    int S[maxn],top;
    int value[maxn];//to_lower
    void tarjan(int x,int fa) {
        low[x]=dfn[x]=++tot; vis[x]=1;
        value[x]=1;
        for (int i=head[x]; ~i; i=edge[i].next) {
            int v=edge[i].to;
            if (used[edge[i].id]) continue;
            if (v==fa) continue;
            S[++top]=edge[i].id;
            used[edge[i].id]=1;
            if (!dfn[v]) {
                tarjan(v,x);
                low[x]=min(low[x],low[v]);
                if (dfn[x]<=low[v]) { //割边和边双联通
                    vector<int> eid,pid,basp;
                    int nowid=x;
                    while (1) {
                        int id=S[top--];
                        eid.push_back(id);
                        nowid^=e[id].u^e[id].v;
                        pid.push_back(nowid);//last
                        basp.push_back(1);//msut; 得正反分
别一遍
                        debug("mapping: %d:
(%d,%d)\n",nowid,e[id].u,e[id].v);
                        if (id==edge[i].id) break;
                    } if (low[v]==dfn[x]) { //双联通, 在这里 dp
                        deputs(" circle :");//环, eid; pid
                        ll nowp=1;
                        for (int _=0; _<(int)eid.size()-1; _++)
{//正
                            int noweid=eid[_];
                            edges nowe=e[noweid];
                            nowp=nowp*nowe.len%M;
                            basp[_]=nowp;
                            debug("e:
(%d-%d) : %d\n",nowe.u,nowe.v,nowe.len);
                        } nowp=1;
                        for (int _=eid.size()-2; _>=0; _--) {//
倒着
                            int noweid=eid[_+1];
                            edges nowe=e[noweid];
                            nowp=nowp*nowe.len%M;
                            ll
base=M+1-(M+1-nowp)*(M+1-basp[_])%M;//*oth[i]

add_(value[x],value[pid[_]]*base%M);
```

```
                                        debug("e(2):
(%d-%d) %d : %d\n",nowe.u,nowe.v,pid[_],nowe.len);
                                    }
                                } else {
                                    deputs(" tree :");
                                    for (int _=0; _<(int)eid.size(); _++) {
                                        int noweid=eid[_];
                                        edges nowe=e[noweid];

add_(value[x],value[pid[_]]*nowe.len%M);
                                        debug("e:
(%d-%d) %d : %d\n",nowe.u,nowe.v,pid[_],nowe.len);
                                        // E.push_back(e[now]);//割边
                                    }
                                }
                                debug("%d %d\n",low[v],dfn[x]);
                            }
                        } else if (vis[v])
                            low[x]=min(low[x],dfn[v]);
                    }
                }
                void init(int n,int m) {
                    int i;
                    FOR(i,1,m) used[i]=0;
                    FOR(i,1,n) head[i]=-1,dfn[i]=0; etot=tot=0;
                    FOR(i,1,m)
addedge(e[i].u,e[i].v,i),addedge(e[i].v,e[i].u,i);
                    // FOR(i,1,n) if (!dfn[i]) tarjan(i,0);
                }
            }

            int main() {
                int T,_; T=1;
                scanf("%d",&T);
                FOR(_,1,T) {
                    int n,m,i;
                    scanf("%d%d",&n,&m);
                    FOR(i,1,m) {
                        int a,b;
                        scanf("%d%d%d%d",&e[i].u,&e[i].v,&a,&b);
                        e[i].len=a*powMM(b,M-2)%M;
                    } tarjan::init(n,m);
                    tarjan::tarjan(1,0);
                    ll Ans=tarjan::value[1];
                    printf("Case #%d: %lld\n",_,Ans);
                }
            }
```

环套外向树

```
// wannafly 挑战赛 16E
// 题意: 给个基环内向树, 每个点每时刻走 1
// 问你最后某时刻 某个 pos 有几个点
// 做法是基环内向树 dp 一下, 分两部分贡献算一下
struct node {
    int l,r,val;
} T[maxn*20]; int ntot;
void ins(int &x,int pos,int L,int R) {
    if (!x) x=++ntot; T[x].val++;
    if (L==R) return;
    int mid=(L+R)/2;
    if (pos<=mid) ins(T[x].l,pos,L,mid);
    else ins(T[x].r,pos,mid+1,R);
}
int que(int x,int l,int r,int L,int R) {
    if (!x) return 0;
    if (l<=L&&R<=r) return T[x].val;
    int ret=0,mid=(L+R)/2;
    if (l<=mid) ret+=que(T[x].l,l,r,L,mid);
    if (mid<r) ret+=que(T[x].r,l,r,mid+1,R);
    return ret;
}
int A[maxn];
vector<int> cir,edge[maxn];
map<int,int> cirnum[maxn];
int vis[maxn],cfa[maxn],circnt[maxn],dep[maxn];
int in[maxn],out[maxn],dtot,ctot;
void dfs(int x,int depth,int cir_id) {
    vis[x]=1; in[x]=++dtot; cfa[x]=cir_id; dep[x]=depth;
    for (int v:edge[x]) dfs(v,depth+1,cir_id);
    out[x]=dtot;
}
void solve(int x) {
    cir.clear(); ctot++;
    while (A[x]&&!vis[A[x]]) x=A[x],vis[x]=1;
    while (A[x]&&vis[A[x]]==1) {
        vis[A[x]]=2; cir.push_back(x);
        cfa[x]=ctot; x=A[x];
    } int i; circnt[ctot]=cir.size();
    rREP(i,cir.size()-1) dep[cir[i]]=dep[A[cir[i]]]+1;
    for (int v:cir) for (int y:edge[v]) if (vis[y]!=2) dfs(y,1,v);
}

int n,m;
int root[maxn];
vector<pair<int,int> > t_t[maxn];
void update(int i,int k) {
    if (vis[k]==1) {
        ins(root[i+dep[k]],in[k],1,n);
        i+=dep[k]; k=cfa[k];

t_t[i].push_back(make_pair(cfa[k],(i+dep[k])%circnt[cfa[k]]));
    } else cirnum[cfa[k]][(i+dep[k])%circnt[cfa[k]]]++;
}
int getans(int i,int k) {
    if (vis[k]==1) return que(root[i+dep[k]],in[k],out[k],1,n);
    else return cirnum[cfa[k]][(i+dep[k])%circnt[cfa[k]]];
}
int lastans;
int main() {
    int i,k;
    scanf("%d",&n);
    FOR(i,1,n) {
        scanf("%d",&A[i]);
        edge[A[i]].push_back(i);
    }
    FOR(i,1,n) if (!vis[i]) solve(i);
    scanf("%d",&m);
    FOR(i,1,m) {
        scanf("%d",&k);
        k^=lastans;
        update(i,k);
        for (auto now:t_t[i])
cirnum[now.first][now.second]++;
        lastans=getans(i,k);
        debug(" ans = ");
        printf("%d\n",lastans);
    }
    return 0;
}
```

网络流

最大权闭合图

题意:给定一个有向图,每个点有权值,求最大权闭合图(与没选的没边相连),使得 sigma(val)最大

做法:S->+node(val);-node->T(-val);原边->INF,与 S 相连的最小割即为所求

原因:简单割=>切的全是和 S,T 相连的边

假设最终与 S 相连的点正的 x1,负的 y1;T 的正的 x2,负的 y2,(x2=S 切,y1=T 切)

最小割 C=S 切的正的+T 切的负的=x2+y1(即反过来)

要求的 val=x1-y1

C+val=x1+x2=定值,val=x1+x2-C

C 最小,即最大流

最大密度子图

边数/点数最大

这个是转化成权闭合图的做法：

二分答案

将边看成点

S->边,1

边->连着的两点,1

每个点->T,val

求完即可

因为 边-k*点>=0,二分出这个即可得到答案

做法二：

s->顶点，权值 m

顶点之间连边，权值 1

顶点->T，m+2*ans-d[i](度数)

满流就 OK

最小割的点可以放到边上，然后考虑边！

做法：奇偶染色，拆点然后最小割

最小割填 INF 边的意义：使得一个矩形不可行！

最小路径覆盖:

将原图拆点成两半，然后连成二分图(边就分开来)

然后求个最大匹配(当然跑网络流也行)

要求的路径就是，最大匹配走的路径

答案是边数减去匹配的边数

这里输出方案有个 trick，拓扑排序感觉写起来最舒服

```
//DINIC+当前弧优化
namespace maxflow {
    typedef int type;
    const type INF=0x3f3f3f3f;
    struct node {
        int to; type cap; int next;
        node(int t=0,type c=0,int n=0):to(t),cap(c),next(n) {};
    } edge[maxn*50];
    int head[maxn],tot;
    void addedge(int from,int to,type cap,type rcap=0) {
        edge[tot]=node(to,cap,head[from]);
head[from]=tot++;
        edge[tot]=node(from,rcap,head[to]); head[to]=tot++;
    }
    int dep[maxn],cur[maxn];//当前弧优化
    bool bfs(int s,int t,int n) {
        static int Q[maxn],ST,ED;
        memset(dep+1,0,n*sizeof(int));
        ST=0; ED=-1;
        Q[++ED]=s; dep[s]=1;
        while (ST<=ED) {
            int u=Q[ST++];
            for (int i=head[u]; i!=-1; i=edge[i].next) {
                int v=edge[i].to;
                if (!dep[v]&&edge[i].cap) {
                    Q[++ED]=v; dep[v]=dep[u]+1;
                }
            }
        } return (dep[t]!=0);
    }
    type dfs(int x,const int &t,type flow=INF) {
        if (x==t||flow==0) return flow;
        type ret=0;
        for (int i=cur[x]; i!=-1; i=edge[i].next) {
            if (dep[x]+1==dep[edge[i].to]&&edge[i].cap){
                type f=dfs(edge[i].to,t,min(flow,edge[i].cap));
                edge[i].cap-=f; edge[i^1].cap+=f;
                ret+=f; flow-=f; cur[x]=i;
                if (flow==0) break;
            }
        } if (!ret) dep[x]=0;
        return ret;
    }
    type maxflow(int s,int t,int n) {
        type ret=0;
        while (bfs(s,t,n)) {
            type f;
            memcpy(cur+1,head+1,n*sizeof(int));
            while ((f=dfs(s,t))>0) ret+=f;
        } return ret;
    }
```

```
    void init(int n) {
        memset(head+1,0xff,n*sizeof(int)); tot=0;
    }
}

//ISAP
namespace maxflow {
    typedef LL type;
    const type INF=0x3f3f3f3f3f3f3f3f;
    struct node {
        int to; type cap; int next;
        node(int t=0,type c=0,int n=0):to(t),cap(c),next(n) {};
    } edge[maxn*50];
    int head[maxn],tot;
    void addedge(int from,int to,type cap,type rcap=0) {
        edge[tot]=node(to,cap,head[from]);
head[from]=tot++;
        edge[tot]=node(from,rcap,head[to]); head[to]=tot++;
    }
    int gap[maxn],dep[maxn],cur[maxn];
    void bfs(int s,int t,int n) {//t 好像没啥用啊=_=
        static int Q[maxn],ST,ED;
        memset(dep+1,0xff,n*sizeof(int));
        memset(gap+1,0,n*sizeof(int));
        gap[0]=1; dep[t]=0;
        ST=0; ED=-1; Q[++ED]=t;
        while (ST<=ED) {
            int u=Q[ST++];
            for (int i=head[u]; ~i; i=edge[i].next) {
                int v=edge[i].to;
                if (dep[v]!=-1) continue;
                Q[++ED]=v; dep[v]=dep[u]+1;
                gap[dep[v]]++;
            }
        }
    }
    int S[maxn];
    type sap(int s,int t,int n) {
        bfs(s,t,n);
        memcpy(cur+1,head+1,n*sizeof(int));
        int top=0,u=s; type ret=0;
        while (dep[s]<n) {
            if (u==t) {
                type MIN=INF,inser=0,i;
                REP(i,top) if (MIN>edge[S[i]].cap)
```

```
                    MIN=edge[S[i]].cap,inser=i;
                REP(i,top) {

edge[S[i]].cap-=MIN,edge[S[i]^1].cap+=MIN;
                } ret+=MIN; top=inser; u=edge[S[top]^1].to;
                continue;
            } bool flag=0; int v;
            for (int i=cur[u]; ~i; i=edge[i].next) {
                v=edge[i].to;
                if (edge[i].cap&&dep[v]+1==dep[u]) {
                    flag=1; cur[u]=i; break;
                }
            } if (flag) {
                S[top++]=cur[u]; u=v; continue;
            } int MIN=n;
            for (int i=head[u]; ~i; i=edge[i].next) {
                v=edge[i].to;
                if (edge[i].cap&&dep[v]<MIN)
                    MIN=min(MIN,dep[v]),cur[u]=i;
            } gap[dep[u]]--;
            if (ret>INF) return ret;//not okay
            if (!gap[dep[u]]) return ret;
            dep[u]=MIN+1; gap[dep[u]]++;
            if (u!=s) u=edge[S[--top]^1].to;
        } return ret;
    }
    void init(int n) {
        memset(head+1,0xff,n*sizeof(int)); tot=0;
    }
}
```

无向图全局最小割

无向图 分成两块最小割

做法:O(n^3)|O(nmlogm)

观察到最小割一定是两块中找个点的最小割

那么我们考虑每次找到 S->T 的最小割后缩点

随便找最小割的方法:O(n^2)|O(mlogm)

得到 s,t 的方法:先任意找个 a 开始

定义集合 A:一些点的集合

定义 w(A,v):v 到 A 中所有点的 sum_value

每次从中找出 w 最大的点加入 A

最后加入的两个点记为 S,T

S->T 的最大流的大小为最末的 w

O(nmlogm)

```
bool deleted[maxn],vis[maxn];
vector<pair<int,int> > edge[maxn];
priority_queue<pair<int,int> > Q;
int weight[maxn];
int fa[maxn];
inline int getfa(int x){
    if (fa[x]==x) return x;
    return fa[x]=getfa(fa[x]);
}
int getst(int &s,int &t,int n){
    int i;t=1;
    while (Q.size()) Q.pop();
    REP(i,n-1){
        vis[s=t]=1;
        for (auto &e:edge[s]) {
            int v=getfa(e.second);
            e.second=v;
            if (!vis[v])
                Q.push(make_pair(weight[v]+=e.first,v));
        }t=0;
        while (!t&&Q.size()){
            auto now=Q.top();Q.pop();
            int v=now.second;
            if (!vis[v]) t=v;
        }if (!weight[t]) return 0;
    }return weight[t];
}
int mincut(int n){
    int ret=INF;
    int s,t,i,j,k;
    FOR(i,1,n) deleted[i]=0,fa[i]=i;
    rFOR(i,2,n){
        FOR(j,1,n) weight[j]=0,vis[j]=0;
        ret=min(ret,getst(s,t,i));
        if (!ret) return 0;
        for (auto v:edge[t]) edge[s].push_back(v);
        int x=getfa(s),y=getfa(t);fa[y]=x;
        vector<pair<int,int> >().swap(edge[t]);
    }return ret;
}
```

O(n^3)

```
LL edge[507][507];
bool deleted[maxn],vis[maxn];
vector<int> id;
LL weight[maxn];
LL getst(int &s,int &t,int n){
    int i;t=1;
    for (int v:id) weight[v]=0,vis[v]=0;
    REP(i,n-1){
        vis[s=t]=1;
        for (int v:id) if (!vis[v])
            weight[v]+=edge[s][v],t=v;
        for (int v:id) if (!vis[v])
            if (weight[v]>=weight[t]) t=v;
        if (!weight[t]) return 0;
    }return weight[t];
}
LL mincut(int n){
    LL ret=INFF;
    int s,t,i,j,k;
    FOR(i,1,n) deleted[i]=0;
    rFOR(i,2,n){
        j=0;id.clear();
        FOR(k,1,n) if (!deleted[k]) id.push_back(k);
        ret=min(ret,getst(s,t,id.size()));
        if (!ret) return 0;
        for (int v:id) if (v!=s&&v!=t){
            edge[s][v]+=edge[t][v];
            edge[v][s]+=edge[v][t];
        }deleted[t]=1;
    }return ret;
}
```

无向图最小割树 GH-tree

```
//两点的 LCA_MAX 是最小割
namespace gomoryhu_tree {
    typedef int type;
    struct node { //只能是双向的
        int u,v; type len;
        node(int u=0,int v=0,type len=0):u(u),v(v),len(len) {};
    } edge[maxn],e[maxn];
    int tot,etot;
    void addedge(int u,int v,int len) {
        edge[++tot]=node(u,v,len);
    } int n;
    void solve(int l,int r,int id[]) {//id,id+n
        static int tmp[maxn];
        if (l==r) return;
        random_shuffle(id+l,id+r+1);
            maxflow::init(n); int i,L=l,R=r;
            FOR(i,1,tot)
maxflow::addedge(edge[i].u,edge[i].v,edge[i].len,edge[i].len);

e[++etot]=node(id[l],id[r],maxflow::maxflow(id[l],id[r],n));
            FOR(i,l,r) if (maxflow::dep[id[i]])
                tmp[L++]=id[i]; else tmp[R--]=id[i];
            FOR(i,l,r) id[i]=tmp[i];
            solve(l,R,id); solve(L,r,id);
        }
        void init(int _n) {
            n=_n; tot=etot=0;
            srand(time(0));
        }
    }
```

最小费用流

```
    // 这个好像就是 zkw 费用流
    // 拆点后可以 S 向入连边，出向 T 连边，然后入和出就可以保持动态
平衡!
    // 连边是为了将"获取的"和"使用的"联系起来! 大概意思就是，使用
的流量确定...
    // 注意观察特殊性质
    // 费用流有个"短路"的性质，如果流到这里可能会使得其他的流量减
少，这个好像有点用
    // cf 1014921
    // 题意:每个点可以流无限,费用 value
    // 存在 limit 为 l-r 最多用 x 个
    // 流量费用互换，流量转化为差分约束即可
    namespace mincostflow {
        typedef ll type;
        const type INF=0x3f3f3f3f3f3f3f3fll;
        struct node {
            int to; type cap,cost; int rev;
            node(int t=0,type c=0,type _c=0,int n=0):
                to(t),cap(c),cost(_c),rev(n) {};
        }; vector<node> edge[maxn];
        void addedge(int from,int to,type cap,type cost,type
rcap=0) {

edge[from].push_back(node(to,cap,cost,edge[to].size()));

edge[to].push_back(node(from,rcap,-cost,edge[from].size()-1));
        }
    type dis[maxn];
    bool mark[maxn];
    void spfa(int s,int t,int n) {
        memset(dis+1,0x3f,n*sizeof(type));
        memset(mark+1,0,n*sizeof(bool));
        static int Q[maxn],ST,ED;
        dis[s]=0; ST=ED=0; Q[ED++]=s;
        while (ST!=ED) {
            int v=Q[ST]; mark[v]=0;
            if ((++ST)==maxn) ST=0;
            for (node &e:edge[v]) {
                if (e.cap>0&&dis[e.to]>dis[v]+e.cost) {
                    dis[e.to]=dis[v]+e.cost;
                    if (!mark[e.to]) {
                        if (ST==ED||dis[Q[ST]]<=dis[e.to]) {
                            Q[ED]=e.to,mark[e.to]=1;
                            if ((++ED)==maxn) ED=0;
                        } else {
                            if ((--ST)<0) ST+=maxn;
                            Q[ST]=e.to,mark[e.to]=1;
                        }
                    }
                }
            }
        }
    } int cur[maxn];
    type dfs(int x,int t,type flow) {
        if (x==t||!flow) return flow;
        type ret=0; mark[x]=1;
        int i;
        rep(i,cur[x],(int)edge[x].size()) {
            node &e=edge[x][i];
            if (!mark[e.to]&&e.cap) {
                if (dis[x]+e.cost==dis[e.to]) {
                    int f=dfs(e.to,t,min(flow,e.cap));
                    e.cap-=f; edge[e.to][e.rev].cap+=f;
                    ret+=f; flow-=f; cur[x]=i;
                    if (flow==0) break;
                }
            }
        } mark[x]=0;
        return ret;
    }
    pair<type,type> mincostflow(int s,int t,int n,type
flow=INF) {
        type ret=0,ans=0;
        while (flow) {
            spfa(s,t,n); if (dis[t]==INF) break;
            // 这样加当前弧优化会快，我也不知道为啥
            memset(cur+1,0,n*sizeof(int));
            type len=dis[t],f;
            while ((f=dfs(s,t,flow))>0)//while 也行
                ret+=f,ans+=len*f,flow-=f;
        } return make_pair(ret,ans);
    }
    void init(int n) {
        int i; FOR(i,1,n) edge[i].clear();
    }
}
int A[maxn];
int main() {
    int n,m;
    int i;
    scanf("%d%d",&n,&m);
    mincostflow::init(n+1+2);
    int s=n+2,t=n+3;
    FOR(i,1,n) {
        scanf("%d",&A[i]);
        mincostflow::addedge(s,i,A[i],0);
        mincostflow::addedge(i+1,t,A[i],0);
        mincostflow::addedge(i+1,i,INF,0);//i-(i+1)<=0
    }
    FOR(i,1,m) {
        int l,r,c;
        scanf("%d%d%d",&l,&r,&c); r++;
        mincostflow::addedge(l,r,INF,c);//r-l<=n
    }

printf("%lld\n",mincostflow::mincostflow(s,t,n+3,INF).second);
}
//原始对偶 dij,可以跑负边
namespace mincostflow {
    typedef int type;
    const type INF=0x3f3f3f3f;
    struct node {
        int to; type cap,cost; int rev;
        node(int t=0,type c=0,type _c=0,int n=0):
            to(t),cap(c),cost(_c),rev(n) {};
    }; vector<node> edge[maxn];
    void addedge(int from,int to,type cap,type cost,type
rcap=0) {

edge[from].push_back(node(to,cap,cost,edge[to].size()));

edge[to].push_back(node(from,rcap,-cost,edge[from].size()-1));
    }
    int prev[maxn],pree[maxn];//pre_cnt
    type dis[maxn],h[maxn];
    pair<type,type> mincostflow(int s,int t,int n,type
flow=INF) {
            type ret=0,ans=0;
            memset(h+1,0,n*sizeof(type));
            while (flow) {
            // dij
            typedef pair<type,int> pti;
            memset(dis+1,0x3f,n*sizeof(type));
            static
priority_queue<pti,vector<pti>,greater<pti> > Q;
            dis[s]=0; Q.push(pti(0,s));
            while (Q.size()) {
                auto now=Q.top(); Q.pop();
                if (dis[now.second]<now.first) continue;
                int i,v=now.second;
                REP(i,(int)edge[v].size()) {
                    node &e=edge[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(pti(dis[e.to],e.to));
                    }
                }
            } int i;
            if (dis[t]==INF) break;
            FOR(i,1,n) h[i]+=dis[i];
            type d=flow;
            for (int i=t; i!=s; i=prev[i])
                d=min(d,edge[prev[i]][pree[i]].cap);
            if (d==0) break;
            flow-=d; ret+=d; ans+=d*h[t];
            for (int i=t; i!=s; i=prev[i]){
                node &e=edge[prev[i]][pree[i]];
                e.cap-=d; edge[e.to][e.rev].cap+=d;
            }
            } return make_pair(ret,ans);
    }
    void init(int n) {
        int i; FOR(i,1,n) edge[i].clear();
    }
}
```

上下界网络流

```
    //可二分 t->s 边的下/上界,即可达到最大最小流
    //最大流:t->s 连边, ss->tt 流, s->t 正向最大流,会流掉反向建的边的
流量, 流量就是 ans
    //最小流:ss->tt 流, t->s 连边, ss->tt 流, s->t 的就是最大流
    //带权值的直接加权即可, in 和 out 加的边 val=0(只是为了限制流出
可以等于流入而已)
    namespace pipeflow {
        typedef int type;
        int eid[maxn*10],etot;
        type in[maxn],out[maxn],flow[maxn*10];
        int s_s,t_t;//S,T
        int addedge(int u,int v,int low,int high) {
            eid[etot]=maxflow::addedge(u,v,high-low);
            out[u]+=low; in[v]+=low; flow[etot++]=low;
            return etot-1;
        }
        void init(int n) {
            s_s=n+1,t_t=n+2; etot=0;
            memset(in+1,0,n*sizeof(type));
            memset(out+1,0,n*sizeof(type));
            maxflow::init(n+2);
        }
        type solve(int n,int s,int t) {
            int sum=0; int i;
            FOR(i,1,n) {
                sum+=max(0,in[i]-out[i]);
                if (in[i]>out[i])
maxflow::addedge(s_s,i,in[i]-out[i]);
                if (in[i]<out[i])
maxflow::addedge(i,t_t,out[i]-in[i]);
            }
            // // maxflow:
            // maxflow::addedge(t,s,INF);
            // if (maxflow::maxflow(s_s,t_t,n+2)!=sum) return -1;
            // return maxflow::maxflow(s,t,n+2);//maxflow

            // // minflow:
```

```
            // type first=maxflow::maxflow(s_s,t_t,n+2);
            // int retpos=maxflow::addedge(t,s,INF);
            // if (first+maxflow::maxflow(s_s,t_t,n+2)!=sum)
return -1;
            // return maxflow::edge[retpos^1].cap;//minflow

            // okay flow:
            // if (maxflow::maxflow(s_s,t_t,n+2)!=sum) return 0;
            REP(i,etot)
flow[i]+=maxflow::edge[eid[i]^1].cap;//edges
            //return 1;
        }
    }
```

树分治

```
    //HDU6268
    //ccpc2017 杭州
    //树分治后 树形依赖 DP
    int A[maxn];
    vector<int> edge[maxn];
    int sz[maxn];
    bool mark[maxn];
    int minweight,root;
    void dfs1(int x,int fa,int n) {
        int weight=0; sz[x]=1;
        for (int v:edge[x]) {
            if (v==fa||mark[v]) continue;
            dfs1(v,x,n); sz[x]+=sz[v];
            weight=max(weight,sz[v]);
        } weight=max(weight,n-sz[x]);
        if (weight<minweight) root=x,minweight=weight;
    }
    bitset<100007> now[3007],ans;//depth
    void dfs2(int x,int fa,int dep) {
        now[dep]=now[dep-1]; sz[x]=1;
        for (int v:edge[x]) {
            if (v==fa||mark[v]) continue;
            dfs2(v,x,dep+1); sz[x]+=sz[v];
        } now[dep-1]|=now[dep]<<A[x];
    }
    void calc(int x){
        now[0].reset(); now[0].set(0);
        dfs2(x,0,1); ans|=now[0];
    }
    void dfs3(int x) {
        calc(x); mark[x]=1;
        for (int v:edge[x]) {
            if (mark[v]) continue;
            minweight=sz[v];
            dfs1(v,0,sz[v]);
            dfs3(root);
        }
    }
    int main() {
        int n,m,T;
        int i;
        scanf("%d",&T);
        while (T--) {
            scanf("%d%d",&n,&m);
            REP(i,n-1) {
                int u,v;
                scanf("%d%d",&u,&v);
                edge[u].push_back(v);
                edge[v].push_back(u);
            } FOR(i,1,n) scanf("%d",&A[i]);
            minweight=n;
            dfs1(1,0,n); dfs3(root);
            FOR(i,1,m) printf("%d",(int)ans[i]);
            puts("");
            ans.reset();
            FOR(i,1,n) edge[i].clear(),mark[i]=0;
        }
        return 0;
    }
```

动态点分治

```
    //题意：动态查询到某点距离不超过 x 的权值和，更改某点权值
    //注意容斥的时候的 length 位置不是 root~是上个 root 相连的位置
    //也就是说 dis 得单独计算//dfs2 一次比两次少一半多的常数=_=
    //addnode 中是 ids
    int
BIT_pool[maxn*40],*BIT[maxn],*SUBBIT[maxn],*st=BIT_pool;
    int size[maxn]; bool mark[maxn];
    int minweight,root;
    struct Node {
        int to,next;
        Node(int _to=0,int _next=0):to(_to),next(_next) {};
    } edge[maxn*2];
    int head[maxn],tot;
    void addedge(int u,int v) {
        edge[++tot]=Node(v,head[u]); head[u]=tot;
    }
    void dfs1(int x,int fa,int n) {
        int weight=0; size[x]=1;
        for (int i=head[x]; ~i; i=edge[i].next) {
            int v=edge[i].to;
            if (v==fa||mark[v]) continue;
            dfs1(v,x,n);
            size[x]+=size[v];
            weight=max(weight,size[v]);
        } weight=max(weight,n-size[x]);
        if (weight<minweight) {root=x; minweight=weight;}
    }
    int length[maxn];
    struct node {
        int top,sub,len,next;
        node() {}
        node(int _top,int _sub,int _len,int
_next):top(_top),sub(_sub),len(_len),next(_next) {};
    } nodes[maxn*20];
    int calhead[maxn],caltot;
    int maxdep;
    void addnode(int x,int top,int sub,int len) {
        nodes[++caltot]=node(top,sub,len,calhead[x]);
calhead[x]=caltot;
    }
    void dfs2(int x,int fa,int top,int sub,int dep) {
        addnode(x,top,sub,dep);
        for (int i=head[x]; ~i; i=edge[i].next) {
            int v=edge[i].to;
            if (v==fa||mark[v]) continue;
            dfs2(v,x,top,sub,dep+1);
        } maxdep=max(maxdep,dep);
    }
    int len[maxn],sublen[maxn];
    void dfs3(int x) {
        mark[x]=1; root=x;
        maxdep=0; int xdep=0;
        addnode(x,x,0,0);
        for (int i=head[x]; ~i; i=edge[i].next) {
            int v=edge[i].to;
            if (mark[v]) continue;
            minweight=size[v]; dfs1(v,0,size[v]);
                maxdep=0; dfs2(v,0,x,root,1); //判重是 x,init_dep=1
                sublen[root]=maxdep; xdep=max(xdep,maxdep);
                SUBBIT[root]=st; st+=sublen[root]+1;
                dfs3(root);
            } len[x]=xdep;
            BIT[x]=st; st+=len[x]+1;
        }
        inline int lowbit(int x) {return x&-x;}
        void add(int *T,int n,int x,int val) {
            x++; T--; n++;
            for (; x<=n; x+=lowbit(x)) T[x]+=val;
        } int get(int *T,int x) {
            x++; T--; int ret=0;
            for (; x; x-=lowbit(x)) ret+=T[x];
            return ret;
        }
        void update(int x,int val) {
            for (int i=calhead[x]; ~i; i=nodes[i].next) {
                int v=nodes[i].top,length=nodes[i].len;
                add(BIT[v],len[v],length,val);
                v=nodes[i].sub;
                if (v) add(SUBBIT[v],sublen[v],length,val);
            }
        } int query(int x,int dis) {
            int ret=0;
            for (int i=calhead[x]; ~i; i=nodes[i].next) {
                int v=nodes[i].top,length=nodes[i].len;
                if (dis>=length) {
                    ret+=get(BIT[v],min(dis-length,len[v]));
                    v=nodes[i].sub;
                    if (v)
ret-=get(SUBBIT[v],min(dis-length,sublen[v]));;
                }
            } return ret;
        }
        int n,m,T;
        int i,j,k;
        char op[2];
        int a[maxn];
        int main() {
            while (~scanf("%d%d",&n,&m)) {
                FOR(i,1,n) mark[i]=0,BIT[i]=SUBBIT[i]=nullptr;
                memset(BIT_pool,0,sizeof(int)*(st-BIT_pool));
st=BIT_pool;
                FOR(i,1,n) head[i]=calhead[i]=-1; tot=caltot=0;
            FOR(i,1,n) scanf("%d",&a[i]);
            FOR(i,1,n-1) {
                int u,v;
                scanf("%d%d",&u,&v);
                addedge(u,v); addedge(v,u);
            }
            minweight=INF; dfs1(1,0,n);
            dfs3(root);
            FOR(i,1,n) update(i,a[i]);
            FOR(i,1,m) {
                int u,v;
                scanf("%s%d%d",op,&u,&v);
                if (op[0]=='!') update(u,v-a[u]),a[u]=v;
                else printf("%d\n",query(u,v));
            }
        }
    }
```

部分树上 dp

从求含某条边的最小生成树截下来的代码(当然前面 sort 了)合并(要记得 merge 咋写),先 sort 然后从小到大讨论,连 father,之后 merge

```
inline int Union(int u,int v,int len) {
    int ret=0;
    while (u!=v&&(fa[u]!=u||fa[v]!=v)) {
        if (fa[u]==u||fa[v]!=v&&sz[u]>sz[v])
        {ret=max(ret,val[v]); v=fa[v];}
        else {ret=max(ret,val[u]); u=fa[u];}
    } if (u==v) return ret;
    if (sz[u]>sz[v]) swap(u,v);
    fa[u]=v; val[u]=len;
    sz[v]+=sz[u]; ans=ans+len;
    return len;
}
```

2-sat

//重点是维护拆点后各种限制之间的关系，这个是个二分以后 2-sat 的

```
struct T_SAT {
    struct enode {
        int to,next;
        enode(int _to=0,int _next=-1):to(_to),next(_next) {};
    } edge[maxn*maxn*2];
    int head[maxn*2],etot;
    void addedge(int u,int v) {
        edge[++etot]=enode(v,head[u]);
        head[u]=etot;
    }
    int dfn[maxn*2],low[maxn*2],belong[maxn*2];
    bool vis[maxn*2];
    int tot,cnt;
    int S[maxn*2],top;
    void dfs(int x) {
        dfn[x]=low[x]=++tot;
        S[++top]=x;
        vis[x]=1;
        for (int i=head[x]; ~i; i=edge[i].next) {
            int v=edge[i].to;
            if (!dfn[v]) {
                dfs(v);
                low[x]=min(low[x],low[v]);
            } else if (vis[v])
                low[x]=min(low[x],dfn[v]);
        }
        if (dfn[x]==low[x]) {
            cnt++;
            while (1) {
                int now=S[top--];
                vis[now]=0;
                belong[now]=cnt;
                if (now==x) break;
            }
        }
    }
    void init(int n) {
        int i;
        REP(i,2*n) head[i]=-1;
        etot=0;
    }
    bool solve(int n) {
        int i;
        tot=cnt=0;
        REP(i,2*n) dfn[i]=vis[i]=0;
        REP(i,2*n) if (!dfn[i]) dfs(i);
        REP(i,n) if (belong[i]==belong[i+n]) return 0;
        return 1;
    }
} two_sat;
int n,m;
int i,j;
int a1,a2,c1,c2;
int main() {
	while (~scanf("%d%d",&n,&m)) {
		two_sat.init(n);
		REP(i,m) {
			scanf("%d%d%d%d",&a1,&a2,&c1,&c2);
			if (c1==1&&c2==1) {
				two_sat.addedge(a1+n,a2);
				two_sat.addedge(a2+n,a1);
			} else if (c1==0&&c2==1) {
				two_sat.addedge(a1,a2);
				two_sat.addedge(a2+n,a1+n);
			} else if (c1==1&&c2==0) {
				two_sat.addedge(a1+n,a2+n);
				two_sat.addedge(a2,a1);
			} else if (c1==0&&c2==0) {
				two_sat.addedge(a1,a2+n);
				two_sat.addedge(a2,a1+n);
			}
		}
		if (two_sat.solve(n)) puts("YES");
		else puts("NO");
	}
}
```

可持久化的 2-sat 输出方案

```
//对于一般点是对称的题目，直接 belong[i]<belong[i+n]输出即可
//否则需要拓扑排序，破坏了本身良好的性质
// 题意：给颗树，每次给俩路径
// 问你 m 组询问，从每个里选个路径，是否可以不相交
// 做法：可持久化建线段树然后 2-sat
// 输出方案需要把每个块都拓扑排序
namespace T_SAT {
	const static int maxn=5e6+7;
	struct enode {
		int to,next;
		enode(int _to=0,int _next=-1):to(_to),next(_next) {};
	} edge[maxn*6];
	int head[maxn],etot;
	void addedge(int u,int v) {
		edge[++etot]=enode(v,head[u]); head[u]=etot;
	}
	int dfn[maxn],low[maxn],belong[maxn];
	bool vis[maxn];
	int tot,cnt;
	int S[maxn],top;
	void dfs(int x) {
		dfn[x]=low[x]=++tot;
		S[++top]=x; vis[x]=1;
		for (int i=head[x]; ~i; i=edge[i].next) {
			int v=edge[i].to;
			if (!dfn[v]) {
				dfs(v);
				low[x]=min(low[x],low[v]);
			} else if (vis[v])
				low[x]=min(low[x],dfn[v]);
		}
		if (dfn[x]==low[x]) {
			cnt++;
			while (1) {
				int now=S[top--];
				vis[now]=0; belong[now]=cnt;
				if (now==x) break;
			}
		}
	}
	void init() {
		memset(head,-1,sizeof(head)); etot=0;
	}
	void solve(int n) {
		int i; tot=cnt=0;
		FOR(i,1,n) dfn[i]=vis[i]=0;
		FOR(i,1,n) if (!dfn[i]) dfs(i);
	}
}
int choose,remain;
int upid[maxn*8],downid[maxn*8],tot;
void build(int x,int L,int R) {
	upid[x]=++tot; downid[x]=++tot;
	if (downid[x>>1]) {
		T_SAT::addedge(downid[x>>1],downid[x]);
	} if (L==R) return;
	int mid=(L+R)/2;
	build(x<<1,L,mid);
	build(x<<1|1,mid+1,R);
}
bool update;
void query(int x,int l,int r,int L,int R) {
        if (l>r) return;
        if (l<=L&&R<=r) {
            if (!update) {
                T_SAT::addedge(choose,downid[x]);
            } else {
                T_SAT::addedge(++tot,downid[x]);
downid[x]=tot;
                T_SAT::addedge(upid[x],remain);
                T_SAT::addedge(downid[x],remain);
                int
fa=downid[x>>1],ls=downid[x<<1],rs=downid[x<<1|1];
                if (fa) T_SAT::addedge(fa,downid[x]);
                if (ls) T_SAT::addedge(downid[x],ls);
                if (rs) T_SAT::addedge(downid[x],rs);
            }
            return;
        } else if (!update) T_SAT::addedge(choose,upid[x]);
        int mid=(L+R)/2;
        if (l<=mid) query(x<<1,l,r,L,mid);
        if (mid<r) query(x<<1|1,l,r,mid+1,R);
    }
    namespace PRE_CAL {
        vector<int> edge[maxn];
        int fa[maxn],son[maxn],id[maxn],tot;
        int sz[maxn],top[maxn],dep[maxn];
        void dfs_1(int u,int father,int depth) {
            fa[u]=father; dep[u]=depth;
            int mx=-1; sz[u]=1; son[u]=0;
            for (int v:edge[u]) {
                if (father==v) continue;
                dfs_1(v,u,depth+1);
                sz[u]+=sz[v];
                if (sz[v]>mx) mx=sz[v],son[u]=v;
            }
        }
        void dfs_2(int u,int x) {
            id[u]=++tot; top[u]=x;
            if (son[u]) dfs_2(son[u],x);
            for (int v:edge[u]) {
                if (v==fa[u]||v==son[u]) continue;;
                dfs_2(v,v);
            }
        }
        void solve(int x,int y) {
            while (top[x]!=top[y]) {
                if (dep[top[x]]<dep[top[y]]) swap(x,y);
                query(1,id[top[x]],id[x],1,n); x=fa[top[x]];
            } if (dep[x]>dep[y]) swap(x,y);
            if (son[x]) query(1,id[son[x]],id[y],1,n);
        }
    }
    int chosen[maxn];
    int A[maxn],B[maxn],C[maxn],D[maxn];
    int TaskA() {
        int i,j,m;
        FOR(i,1,n-1) {
            int u,v;
            scanf("%d%d",&u,&v);
            PRE_CAL::edge[u].push_back(v);
            PRE_CAL::edge[v].push_back(u);
        } scanf("%d",&m);
        T_SAT::init();
        PRE_CAL::dfs_1(1,0,0);
        PRE_CAL::dfs_2(1,1);
        FOR(i,1,m) chosen[i]=++tot,++tot;
        build(1,1,n);
        FOR(i,1,m) scanf("%d%d%d%d",&A[i],&B[i],&C[i],&D[i]);
        FOR(i,1,m) {
            choose=chosen[i]; remain=chosen[i]+1;
            update=0;
            PRE_CAL::solve(A[i],B[i]);
            swap(choose,remain);
            PRE_CAL::solve(C[i],D[i]);
            update=1; swap(choose,remain);
            PRE_CAL::solve(A[i],B[i]);
            swap(choose,remain);
            PRE_CAL::solve(C[i],D[i]);
        } build(1,1,n);
        rFOR(i,1,m) {
            choose=chosen[i]; remain=chosen[i]+1;
            update=0;
            PRE_CAL::solve(A[i],B[i]);
            swap(choose,remain);
            PRE_CAL::solve(C[i],D[i]);
            update=1; swap(choose,remain);
            PRE_CAL::solve(A[i],B[i]);
            swap(choose,remain);
            PRE_CAL::solve(C[i],D[i]);
        }
        T_SAT::solve(tot);
        FOR(i,1,m) if
(T_SAT::belong[chosen[i]]==T_SAT::belong[chosen[i]+1]) return
0*puts("NO");
        puts("YES");
        FOR(i,1,m)
printf("%d\n",((T_SAT::belong[chosen[i]]<T_SAT::belong[chosen[i]
+1])^1)+1);
        return 0;
    }
```

dfs 序_换根的讨论

```
    //http://codeforces.com/contest/916/problem/E
    //改根,子树加,查,令人窒息的讨论
    //有套路是 dfs 同时更新 value 啥的，需要注意
    LL sum[maxn<<2],lazy[maxn<<2];
    void update(int x,int l,int r,LL val,int L,int R) {
        if (l>r) return;
        if (l<=L&&R<=r) {lazy[x]+=val; sum[x]+=(R-L+1)*val;
return;}
        int mid=(L+R)/2;
        if (lazy[x]) {
            lazy[x<<1]+=lazy[x];
            lazy[x<<1|1]+=lazy[x];
            sum[x<<1]+=(mid-L+1)*lazy[x];
            sum[x<<1|1]+=(R-mid)*lazy[x];
            lazy[x]=0;
        } if (l<=mid) update(x<<1,l,r,val,L,mid);
        if (mid<r) update(x<<1|1,l,r,val,mid+1,R);
        sum[x]=sum[x<<1]+sum[x<<1|1];
    }
    LL query(int x,int l,int r,int L,int R) {
        LL ret=0;
        if (l>r) return 0;
        if (l<=L&&R<=r) return sum[x];
        int mid=(L+R)/2;
        if (lazy[x]) {
            lazy[x<<1]+=lazy[x];
            lazy[x<<1|1]+=lazy[x];
            sum[x<<1]+=(mid-L+1)*lazy[x];
            sum[x<<1|1]+=(R-mid)*lazy[x];
            lazy[x]=0;
        } if (l<=mid) ret+=query(x<<1,l,r,L,mid);
        if (mid<r) ret+=query(x<<1|1,l,r,mid+1,R);
        sum[x]=sum[x<<1]+sum[x<<1|1];
        return ret;
    }
    vector<int> edge[maxn];
    int fa[maxn][27];
    int in[maxn],out[maxn],tot,dep[maxn];
    void dfs(int x,int f,int d) {
        int i;
        fa[x][0]=f; in[x]=++tot; dep[x]=d;
        rep(i,1,20) fa[x][i]=fa[fa[x][i-1]][i-1];
        for (int v:edge[x]) if (v!=f) dfs(v,x,d+1);
        out[x]=tot;
    }
    int lca(int x,int y) {
        int i;
        if (dep[x]<dep[y]) swap(x,y);
        rREP(i,20) if (dep[x]-dep[y]>=1<<i) x=fa[x][i];
        if (x==y) return x;
        rREP(i,20) if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
        return fa[x][0];
    }
    int getnthfa(int x,int k) {
        int i;
        rREP(i,20) if ((k>>i)&1) x=fa[x][i];
        return x;
    }
    int root;
    int n,m;
    int a[maxn];
    int main() {
        int i,j;
        scanf("%d%d",&n,&m);
        FOR(i,1,n) scanf("%d",&a[i]);
        FOR(i,1,n-1) {
            int u,v;
            scanf("%d%d",&u,&v);
            edge[u].push_back(v);
            edge[v].push_back(u);
        }
        dfs(1,0,0);
        FOR(i,1,n) update(1,in[i],in[i],a[i],1,n); root=1;
        while (m--) {
            int op,u,v,x;
            scanf("%d",&op);
            if (op==1) {
                scanf("%d",&root);
        } else if (op==2) {
            scanf("%d%d%d",&u,&v,&x);
            int f=lca(u,v)^lca(v,root)^lca(u,root);
            if (f==root) update(1,1,n,x,1,n);
            else if (lca(f,root)==f) {
                int t=getnthfa(root,dep[root]-dep[f]-1);
                update(1,1,in[t]-1,x,1,n);
                update(1,out[t]+1,n,x,1,n);
            } else update(1,in[f],out[f],x,1,n);
        } else if (op==3) {
            int x;
            LL ans;
            scanf("%d",&x);
            if (x==root) ans=query(1,1,n,1,n);
            else if (in[x]<=in[root]&&in[root]<=out[x]) {
                int t=getnthfa(root,dep[root]-dep[x]-1);

ans=query(1,1,in[t]-1,1,n)+query(1,out[t]+1,n,1,n);
            } else ans=query(1,in[x],out[x],1,n);
            printf("%I64d\n",ans);
        }
    }
}
```

01 序列转树

```
    FOR(i,1,n) {
        scanf("%d",&b[i]);
        if (b[i]) {//up
            pid[i]=now;
            if (!fa[now]) {
                root=++tot_id; fa[now]=tot_id;
                edge[tot_id].push_back(now);
            } now=fa[now];
        } else {//down
            fa[++tot_id]=now;
            edge[now].push_back(tot_id);
            now=tot_id; pid[i]=now;
        }
    }
```

有方向的树链剖分

```
int a[maxn],tot;
int mxr[maxn<<2],mxl[maxn<<2];
int mx[maxn<<2],mn[maxn<<2];
int lazy[maxn<<2];//profit=mx-mn
void change(int x,int val) {
    lazy[x]+=val;
    mx[x]+=val; mn[x]+=val;
}
void pushup(int x) {

mxr[x]=max(max(mxr[x<<1],mxr[x<<1|1]),mx[x<<1|1]-mn[x<<1]
);//->

mxl[x]=max(max(mxl[x<<1],mxl[x<<1|1]),mx[x<<1]-mn[x<<1|1]);
//<-
    mx[x]=max(mx[x<<1],mx[x<<1|1]);
    mn[x]=min(mn[x<<1],mn[x<<1|1]);
}
void pushdown(int x) {
    if (lazy[x]) {
        change(x<<1,lazy[x]);
        change(x<<1|1,lazy[x]);
        lazy[x]=0;
    }
}
void build(int x,int l,int r) {
    mxr[x]=mxl[x]=mx[x]=mn[x]=lazy[x]=0;
    if (l==r) {
        mx[x]=mn[x]=a[l];
        return;
    } int mid=(l+r)/2;
    build(x<<1,l,mid);
    build(x<<1|1,mid+1,r);
    pushup(x);
}
int query(int x,int l,int r,bool flag,int &vmin,int &vmax,int
L,int R,int val) { //flag:-> (top->bottom yes)
    if (l<=L&&R<=r) {
        change(x,val);
        vmin=mn[x]; vmax=mx[x];
        return flag?mxr[x]:mxl[x];
    } pushdown(x);
    int
mid=(L+R)/2,ret=0,mx1=-INF,mx2=-INF,mn1=INF,mn2=INF;
    if (mid>=l)
ret=max(ret,query(x<<1,l,r,flag,mn1,mx1,L,mid,val));
    if (r>mid)
ret=max(ret,query(x<<1|1,l,r,flag,mn2,mx2,mid+1,R,val));
        if (flag) ret=max(ret,mx2-mn1);
        else ret=max(ret,mx1-mn2);
        vmax=max(mx1,mx2);
        vmin=min(mn1,mn2);
        pushup(x);
        return ret;
    }
    int n,q; int i,j,k;
    int u,v,val;
    int b[maxn];
    vector<int> edge[maxn];
    int
sz[maxn],fa[maxn],dep[maxn],son[maxn],top[maxn],id[maxn];
    void dfs1(int u,int from,int depth) {
        int v,i,mx=-1;
        sz[u]=1; fa[u]=from; dep[u]=depth; son[u]=0;
        REP(i,edge[u].size()) {
            v=edge[u][i];
            if (v==from) continue;
            dfs1(v,u,depth+1);
            sz[u]+=sz[v];
            if (sz[v]>mx) mx=sz[v],son[u]=v;
        }
    }
    void dfs2(int u,int x) {
        int v,i;
        top[u]=x; id[u]=++tot;
        if (son[u]) dfs2(son[u],x);
        REP(i,edge[u].size()) {
            v=edge[u][i];
            if (v==fa[u]||v==son[u]) continue;
            dfs2(v,v);
        }
    }
    int Query(int x,int y,int val) {
        int
ret=0,mxx=-INF,mnx=INF,mxy=-INF,mny=INF,vmax,vmin;
        while (top[x]!=top[y]) {
            if (dep[top[x]]>dep[top[y]]) {

ret=max(ret,query(1,id[top[x]],id[x],0,vmin,vmax,1,tot,val));
                ret=max(ret,vmax-mnx);
                mxx=max(mxx,vmax); mnx=min(mnx,vmin);
                x=fa[top[x]];
            } else {

ret=max(ret,query(1,id[top[y]],id[y],1,vmin,vmax,1,tot,val));
                ret=max(ret,mxy-vmin);
                mxy=max(mxy,vmax); mny=min(mny,vmin);
                y=fa[top[y]];
            }
        }
        if (dep[x]>dep[y]) {

ret=max(ret,query(1,id[y],id[x],0,vmin,vmax,1,tot,val));
            ret=max(ret,vmax-mnx);
            mxx=max(mxx,vmax); mnx=min(mnx,vmin);
        } else {

ret=max(ret,query(1,id[x],id[y],1,vmin,vmax,1,tot,val));
            ret=max(ret,mxy-vmin);
            mxy=max(mxy,vmax); mny=min(mny,vmin);
        } ret=max(ret,mxy-mnx);
        return ret;
    }
    int T;
    int main() {
        scanf("%d",&T);
        while (T--) {
            scanf("%d",&n);
            FOR(i,1,n) scanf("%d",&b[i]);
            FOR(i,1,n) edge[i].clear();
            FOR(i,1,n-1) {
                scanf("%d%d",&u,&v);
                edge[u].push_back(v);
                edge[v].push_back(u);
            }
            tot=0;
            dfs1(1,0,1);
            dfs2(1,1);
            FOR(i,1,tot) a[id[i]]=b[i];
            build(1,1,tot);
            scanf("%d",&q);
            REP(i,q) {
                scanf("%d%d%d",&u,&v,&val);
                printf("%d\n",Query(u,v,val));
            }
        }
    }
```

轻重儿子分开维护

```
// 题意：更改链上的边 col
// 更改某个链相邻的边 col
// 查询黑点数
// 做法：轻重边分开维护
struct segment_tree {
    int val[maxn<<2],len[maxn<<2],lazy[maxn<<2];
    void build(int x,int L,int R) {
        len[x]=R-L+1; val[x]=0; lazy[x]=0;
        if (L==R) return;
        int mid=(L+R)/2;
        build(x<<1,L,mid);
        build(x<<1|1,mid+1,R);
    }
    void Inverse(int x) {
        lazy[x]^=1; val[x]=len[x]-val[x];
    }
    void pushdown(int x) {
        if (lazy[x]) {
            Inverse(x<<1);
            Inverse(x<<1|1);
            lazy[x]=0;
        }
    }
    void pushup(int x) {
        val[x]=val[x<<1]+val[x<<1|1];
    }
    void update(int x,int l,int r,int L,int R) {
        debug("update: %d %d %d\n",x,l,r);
        if (l<=L&&R<=r) {Inverse(x); return;}
        int mid=(L+R)/2;
        pushdown(x);
        if (l<=mid) update(x<<1,l,r,L,mid);
        if (mid<r) update(x<<1|1,l,r,mid+1,R);
        pushup(x);
    }
    int query(int x,int l,int r,int L,int R) {
        if (l<=L&&R<=r) return val[x];
        int mid=(L+R)/2,ret=0;
        pushdown(x);
        if (l<=mid) ret+=query(x<<1,l,r,L,mid);
        if (mid<r) ret+=query(x<<1|1,l,r,mid+1,R);
        pushup(x);
        return ret;
    }
} heavy,light;
vector<int> edge[maxn];
int fa[maxn],dep[maxn],sz[maxn],tot;
int top[maxn],id[maxn],son[maxn];
void dfs1(int u,int father,int depth) {
    int mx=-1; sz[u]=1;
    fa[u]=father; son[u]=0; dep[u]=depth;
    for (int v:edge[u]) {
        if (v==father) continue;
        dfs1(v,u,depth+1); sz[u]+=sz[v];
        if (sz[v]>mx) mx=sz[v],son[u]=v;
    }
}
void dfs2(int u,int x) {
    top[u]=x; id[u]=++tot;
    if (son[u]) dfs2(son[u],x);
    for (int v:edge[u]) {
        if (v==fa[u]||v==son[u]) continue;
        dfs2(v,v);
    }
}
inline void InverseEdge(int x,int y) {
    while (top[x]!=top[y]) {
        if (dep[top[x]]<dep[top[y]]) swap(x,y);
        heavy.update(1,id[top[x]],id[x],1,n);
        x=fa[top[x]];
    }
    if (dep[x]>dep[y]) swap(x,y);
    if (son[x]) heavy.update(1,id[son[x]],id[y],1,tot);
}
inline void InverseNode(int x,int y) {
    while (top[x]!=top[y]) {
        if (dep[top[x]]<dep[top[y]]) swap(x,y);
        light.update(1,id[top[x]],id[x],1,n);
        heavy.update(1,id[top[x]],id[top[x]],1,n);
        if (son[x]) heavy.update(1,id[son[x]],id[son[x]],1,n);
        x=fa[top[x]];
    }
    if (dep[x]>dep[y]) swap(x,y);
    light.update(1,id[x],id[y],1,tot);
    heavy.update(1,id[x],id[x],1,n);
    if (son[y]) heavy.update(1,id[son[y]],id[son[y]],1,n);
}
inline int Query(int x,int y) {
        int ret=0;
        while (top[x]!=top[y]) {
            if (dep[top[x]]<dep[top[y]]) swap(x,y);
            if (top[x]!=x)
ret+=heavy.query(1,id[son[top[x]]],id[x],1,n);

ret+=heavy.query(1,id[top[x]],id[top[x]],1,n)^light.query(1,id[fa[
top[x]]],id[fa[top[x]]],1,n);
            x=fa[top[x]];
        }
        if (dep[x]>dep[y]) swap(x,y);
        if (son[x]) ret+=heavy.query(1,id[son[x]],id[y],1,n);
        return ret;
    }
    int TaskA() {
        int i;
        scanf("%d",&n); tot=0;
        FOR(i,1,n) edge[i].clear();
        FOR(i,1,n-1) {
            int u,v;
            scanf("%d%d",&u,&v);
            edge[u].push_back(v);
            edge[v].push_back(u);
        } dfs1(1,0,0); dfs2(1,1);
        heavy.build(1,1,n);
        light.build(1,1,n);
        scanf("%d",&q);
        REP(i,q) {
            int op,u,v;
            scanf("%d%d%d",&op,&u,&v);
            if (op==1) InverseEdge(u,v);
            if (op==2) InverseNode(u,v);
            if (op==3) printf("%d\n",Query(u,v));
        }
        return 0;
    }
```

链分治，动态维护树上 dp

```
    // f[x]:this_ans=max(g[x]+f[heavy],0)
    // g[x]:light_ans=A[x]+sigma{f[light]}
    // w[x]:dp[heavy_son]
    // 把轻链和重链分开维护，在重链上一个序列上 DP
    // 题意是更改某点值，查询联通块的最大权重和
    struct heap {
        multiset<ll> S;
        inline void ins(ll x) {
            S.insert(x);
        }
        inline void del(ll x) {
            multiset<ll>::iterator it=S.lower_bound(x);
            if (it!=S.end()) S.erase(it);
        }
        inline ll top() {
            if (!S.size()) return 0;
            return *S.rbegin();
        }
    } SON[maxn]; // light
    vector<int> edge[maxn];
    int fa[maxn],dep[maxn],sz[maxn],tot;
    int top[maxn],id[maxn],rid[maxn],son[maxn],leaf[maxn];
    void dfs1(int u,int father,int depth) {
        int mx=-1,i; sz[u]=1;
        fa[u]=father; son[u]=0; dep[u]=depth;
        REP(i,(int)edge[u].size()) {
            int v=edge[u][i];
            if (v==father) continue;
            dfs1(v,u,depth+1); sz[u]+=sz[v];
            if (sz[v]>mx) mx=sz[v],son[u]=v;
        }
    }
    int A[maxn];
    // f[x]:this_ans=max(g[x]+f[heavy],0)
    // g[x]:light_ans=A[x]+sigma{f[light]}
    // w[x]:dp[heavy_son]
    ll f[maxn],g[maxn],w[maxn];
    void dfs2(int u,int x) {
        top[u]=x; id[u]=++tot; rid[tot]=u;
        g[u]=A[u]; f[u]=0; int i;
        if (son[u]) dfs2(son[u],x);
        REP(i,(int)edge[u].size()) {
            int v=edge[u][i];
            if (v==fa[u]||v==son[u]) continue;
            dfs2(v,v); SON[u].ins(w[v]);
            g[u]+=f[v]; max_(w[u],w[v]);
        } if (son[u]) {
            leaf[u]=leaf[son[u]];
            max_(f[u],g[u]+f[son[u]]);
            max_(w[u],w[son[u]]);
        } else leaf[u]=u;
        max_(f[u],g[u]); max_(w[u],f[u]);
    }
    struct node {
        ll ls,rs,sum,ans;
        node(ll val=0) {sum=val; ls=rs=ans=max(0ll,val);}
    } T[maxn<<2];
    node merge(const node &A,const node &B) {
        node ret;
        ret.ls=max(A.ls,A.sum+B.ls);
        ret.rs=max(B.rs,B.sum+A.rs);
        ret.ans=max(A.ans,B.ans);
        ret.ans=max(ret.ans,A.rs+B.ls);
        ret.sum=A.sum+B.sum;
        return ret;
    }
    // f[x]:this_ans=max(g[x]+f[heavy],0)
    // g[x]:light_ans=A[x]+sigma{f[light]}
    void build(int x,int L,int R) {
        if (L==R) {
            T[x]=node(g[rid[L]]);
            max_(T[x].ans,SON[rid[L]].top());
            return;
        } int mid=(L+R)/2;
        build(x<<1,L,mid);
        build(x<<1|1,mid+1,R);
        T[x]=merge(T[x<<1],T[x<<1|1]);
    }
    void update(int x,int pos,int L,int R) {
        if (L==R) {
            T[x]=node(g[rid[L]]);
            max_(T[x].ans,SON[rid[L]].top());
            return;
        } int mid=(L+R)/2;
        if (pos<=mid) update(x<<1,pos,L,mid);
        if (mid<pos) update(x<<1|1,pos,mid+1,R);
        T[x]=merge(T[x<<1],T[x<<1|1]);
    }
    node query(int x,int l,int r,int L,int R) {
        if (l<=L&&R<=r) return T[x];
        int mid=(L+R)/2;
        if (r<=mid) return query(x<<1,l,r,L,mid);
        if (mid<l) return query(x<<1|1,l,r,mid+1,R);
        return
merge(query(x<<1,l,r,L,mid),query(x<<1|1,l,r,mid+1,R));
    }
    inline void Update(int x,ll y) {
        g[x]-=A[x]; A[x]=y; g[x]+=A[x];
        while (x) {
            update(1,id[x],1,n);
            node nxtval=query(1,id[top[x]],id[leaf[x]],1,n);
            ll initw=w[top[x]]; w[top[x]]=nxtval.ans;
            ll initg=f[top[x]]; f[top[x]]=nxtval.ls;
            x=fa[top[x]];
            if (x) {
                g[x]-=initg;
                g[x]+=nxtval.ls;
                SON[x].del(initw);
                SON[x].ins(nxtval.ans);
            }
        }
    }
    inline ll Query(int x) {
        return query(1,id[x],id[leaf[x]],1,n).ans;
    }
    int main() {
        int i;
        scanf("%d%d",&n,&q); tot=0;
        FOR(i,1,n) scanf("%d",&A[i]);
        FOR(i,1,n) edge[i].clear();
        FOR(i,1,n-1) {
            int u,v;
            scanf("%d%d",&u,&v);
            edge[u].push_back(v);
            edge[v].push_back(u);
        } dfs1(1,0,0); dfs2(1,1);
        FOR(i,1,n) debug("%d ",id[i]); deputs("");
        FOR(i,1,n) debug("%d ",rid[i]); deputs("");
        build(1,1,n);
        REP(i,q) {
            char op[2];
            scanf("%s",op);
            if (op[0]=='M') {
                int x; ll y;
                scanf("%d%lld",&x,&y);
                Update(x,y);
            } else {
                int x;
                scanf("%d",&x);
                printf("%lld\n",Query(x));
            }
        }
        return 0;
    }
```

DSU on tree

//大概意思就是轻儿子记录答案，重儿子不清空，最后把轻儿子的贡献放到重儿子上；如果是基于深度可合并的，长链剖分是O(n)的

```
// CF741D 辣鸡题
// 问你重排能回文的最长串多长
// 直接上就可以了... 看下 dfs 顺序就行了
vector<int> edge[maxn];
int sz[maxn],son[maxn];
void dfs1(int x) {
    int mx=0; sz[x]=1;
    for (int v:edge[x]) {
        dfs1(v); sz[x]+=sz[v];
        if (sz[v]>mx) son[x]=v,mx=sz[v];
    }
}
int A[maxn],dep[maxn];
int ans[maxn],MX[1<<22|7];
map<int,int> MP[maxn];
int Merge(map<int,int> &A,map<int,int> &B,int x) { //B->A
    int ret=0,i;
    for (auto now:B) {
        int p=now.first,l=now.second;
        if (MX[p]) ret=max(ret,MX[p]+l-2*dep[x]);
        REP(i,22) {
            p=now.first^(1<<i);
//              printf("now=%d; p=%d; %d %d %d\n",now.first,p,MX[p],l,dep[x]);
            if (MX[p]) ret=max(ret,MX[p]+l-2*dep[x]);
        }
    }//merge
    for (auto now:B) {
        int p=now.first,l=now.second;
        MX[p]=max(MX[p],l); A[p]=MX[p];
    } map<int,int>().swap(B);
    return ret;
}
void dfs2(int x) {
    for (int v:edge[x]) if (v!=son[x]) {
            dfs2(v); ans[x]=max(ans[x],ans[v]);
            for (auto now:MP[v]) MX[now.first]=0;
        } if (son[x]) {
        dfs2(son[x]); ans[x]=max(ans[x],ans[son[x]]);
    }//cal
    MP[x][A[x]]=dep[x];
    if (son[x]) {
        ans[x]=max(ans[x],Merge(MP[son[x]],MP[x],x));
        swap(MP[x],MP[son[x]]);
    } else MX[A[x]]=dep[x];
    for (int v:edge[x]) if (v!=son[x]) {
            ans[x]=max(ans[x],Merge(MP[x],MP[v],x));
        }
}
int main() {
    int n,i,j,k;
    char c;
    scanf("%d",&n);
    FOR(i,2,n) {
        int fa;
        scanf("%d %c",&fa,&c);
        A[i]=A[fa]^(1<<(c-'a'));
        dep[i]=dep[fa]+1;
        edge[fa].push_back(i);
    } dfs1(1); dfs2(1);
    FOR(i,1,n) printf("%d ",ans[i]);
    return 0;
}
```

LCA

树链剖分：略

Tarjan:

```
vector<int> edge[maxn];
int fa1[maxn],fa2[maxn];
inline int getfa(int *fa,int x) {
    if (fa[x]==x) return x;
    return fa[x]=getfa(fa,fa[x]);
}
int n,m,q,i,k,u,v;
int ans[maxn];
vector<pair<int,int> > Q[maxn];//v,id
void dfs(int x) {
    int i;
    for (int v:edge[x]) {
        dfs(v); fa2[v]=x;
    }
    REP(i,Q[x].size()) {
        if (fa2[Q[x][i].first]!=Q[x][i].first)
            ans[Q[x][i].second]=getfa(fa2,Q[x][i].first);
    }
}
void solve() {
    scanf("%d%d%d",&n,&m,&q);
    FOR(i,1,n) fa1[i]=fa2[i]=i;
    REP(i,m) {
        scanf("%d%d",&u,&v);
        edge[u].push_back(v);
        fa1[v]=u;
    }
    REP(i,q) {
        scanf("%d%d%d",&k,&u,&v);
        if (k==1) {
            if (getfa(fa1,u)!=getfa(fa1,v)) ans[i]=-1;
            else {
                if (u==v) ans[i]=u;
                else {
                    Q[u].push_back(make_pair(v,i));
                    Q[v].push_back(make_pair(u,i));
                }
            }
        } else {
            edge[u].push_back(v);
            fa1[v]=u; ans[i]=0;
        }
    }
    FOR(i,1,n) if (fa1[i]==i) dfs(i);
    FOR(i,1,n) edge[i].clear(),Q[i].clear();
    REP(i,q) if (ans[i]) printf("%d\n",ans[i]);
}
```

倍增

```
int fa[maxn][21];
int n,i,j;
int dep[maxn];
vector<int> edge[maxn];
void dfs(int x,int depth) {
    dep[x]=depth;
    for (int v:edge[x]) dfs(v,depth+1);
}
int lca(int x,int y) {
    int i;
    if (dep[x]<dep[y]) swap(x,y);
    rREP(i,20) if (dep[x]-dep[y]>=1<<i) x=fa[x][i];
    if (x==y) return x;
    rREP(i,20) if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
    return fa[x][0];
}
int dis(int x,int y) {
    return dep[x]+dep[y]-2*dep[lca(x,y)];
}
int kthfa(int x,int k) {
    int i;
    rREP(i,20) if ((k>>i)&1) x=fa[x][i];
    return x;
}
int walk(int x,int y,int d) {
    int f=lca(x,y);
    if (dep[x]-dep[f]>=d) return kthfa(x,d);
    return kthfa(y,dep[x]+dep[y]-2*dep[f]-d);
}
```

虚树 ST 表求 lca

```
// 题意:问最少去掉几个未标记点可以把所有的标记点全分开
// 做法:建虚树然后树上 DP
// 虚树板子,注意:sort 过程可以提到外边去
// 注意，原先有的标记有的时候会到边上，需要特判的，千万不要 if
struct Edges {
    int to; LL len; int next;
    Edges(int _to=0,LL _len=0,int
_next=0):to(_to),len(_len),next(_next) {}
} edge[maxn*2]; int etot;
int head[maxn];
int fa[maxn];
LL uplen[maxn];
int id[maxn],dfn[maxn],idtot;
inline void addedge(int u,int v,LL len) {
    edge[++etot]=Edges(v,len,head[u]); head[u]=etot;
}
namespace LCA {//内部和外部 dfn 不同...
    int dep[maxn]; LL len[maxn];
    int st_dfn[maxn],tot;
    int ST[maxn*2][20];//only L
    void dfs(int x,int f,int d,LL l) {
        int i; dep[x]=d; len[x]=l;
        st_dfn[x]=++tot; ST[tot][0]=x;
        ::id[++idtot]=x; ::dfn[x]=idtot;
        for (i=head[x]; ~i; i=edge[i].next) if (edge[i].to!=f) {
                int v=edge[i].to;
                ::fa[v]=x; ::uplen[v]=edge[i].len;
                dfs(v,x,d+1,l+edge[i].len);
                ST[++tot][0]=x;
            }
    }
    int t_t[maxn*2];
    inline void initST(int n) {
        int i,j;
        FOR(i,1,n*2) t_t[i]=t_t[i>>1]+1;
        FOR(i,1,n*2) {
            rep(j,1,t_t[i]) {
                int u=ST[i][j-1],v=ST[i-(1<<(j-1))][j-1];
                ST[i][j]=dep[u]<dep[v]?u:v;
            }
        }
    }
    inline int lca(int x,int y) {
        x=st_dfn[x]; y=st_dfn[y];
        if (x>y) swap(x,y);
        int t=t_t[y-x+1]-1;
        x=ST[x+(1<<t)-1][t]; y=ST[y][t];
        return dep[x]<dep[y]?x:y;
    }
    inline LL dis(int x,int y) {
        return len[x]+len[y]-2*len[lca(x,y)];
    }
    void init(int n) {
        memset(head+1,0xff,n*sizeof(int));
        etot=idtot=tot=0;
    }
}

namespace vtree {
    int S[maxn],top;
    int pid[maxn],mark[maxn];
    int vid[maxn],vfa[maxn];
    LL vlen[maxn];
    int cmp(int x,int y) {
        return dfn[x]<dfn[y];
    }
    void addedge(int u,int v) {
        vfa[v]=u; vlen[v]=LCA::dis(u,v);
    }
    int m;
    void vbuild(int n) {
        int i; m=0;
        sort(pid+1,pid+1+n,cmp);
        S[top=1]=pid[1];
        mark[pid[1]]=1;
        FOR(i,2,n) {
            int f=LCA::lca(pid[i-1],pid[i]);
            while (top&&LCA::dep[S[top]]>LCA::dep[f]) {
                int v; vid[++m]=v=S[top--];
                if (top&&LCA::dep[S[top]]>LCA::dep[f])
addedge(S[top],v);
                else addedge(f,v);
            } if (!top||S[top]!=f) S[++top]=f;
            S[++top]=pid[i]; mark[pid[i]]=1;
        } while (top-1)
addedge(S[top-1],S[top]),vid[++m]=S[top--];
        vid[++m]=S[1];
        reverse(vid+1,vid+m+1);
    }
    void vclear() {
        int i;
        FOR(i,1,m) mark[vfa[vid[i]]]=0;
        FOR(i,1,m) mark[vid[i]]=0;
    }
}

int ans;
int cnt[maxn];
void solve() {
    int i;
    FOR(i,1,vtree::m) cnt[vtree::vid[i]]=0;
    rFOR(i,1,vtree::m) {
        int x=vtree::vid[i];
        if (vtree::mark[x]) ans+=cnt[x],cnt[x]=1;
        else if (cnt[x]>1) ans++,cnt[x]=0;
        if (i>1) cnt[vtree::vfa[x]]+=cnt[x];
    }
}
int vis[maxn];
int main() {
    int i;
    int n,q;
    scanf("%d",&n);
    LCA::init(n);
    FOR(i,1,n-1) {
        int u,v;
        scanf("%d%d",&u,&v);
        addedge(u,v,1); addedge(v,u,1);
    } LCA::dfs(1,0,0,0);
    LCA::initST(n);
    scanf("%d",&q);
    while (q--) {
        int m,mark=0;
        scanf("%d",&m);
        FOR(i,1,m) scanf("%d",&vtree::pid[i]);
        FOR(i,1,m) vis[vtree::pid[i]]=1;
        FOR(i,1,m) if (vis[fa[vtree::pid[i]]]) mark=1;
        FOR(i,1,m) vis[vtree::pid[i]]=0;
        if (mark) {puts("-1"); continue;}
        vtree::vbuild(m);
        ans=0; solve();
        vtree::vclear();
        printf("%d\n",ans);
    }
    return 0;
}
```

Ladder 长链剖分 k 级祖先

```
namespace ladder {
    vector<int> edge[maxn];
    int id[maxn]; int tot;
    int
fa[maxn][21],son[maxn],top[maxn],len[maxn],dep[maxn];
    vector<int> ladder[maxn];
    int upp[maxn];
    void dfs(int x,int father=0) {
        fa[x][0]=father; id[++tot]=x;
        for (int v:edge[x]) if (v!=father) dfs(v,x);
    }
    void buildfa() {
        int i,j; dep[id[1]]=0;
        FOR(i,1,tot) rep(j,1,21)
fa[i][j]=fa[fa[i][j-1]][j-1],dep[i]=dep[fa[i][0]]+1;
        rFOR(i,1,tot) {
            int o=0,x=id[i]; top[x]=x;
            ladder[x].clear();
            for (int v:edge[x]) if (v!=fa[x][0]){
                if (!o||len[o]<len[v]) o=v;
            } if (o) len[x]=len[o]+1; else o=0;
            son[x]=o; top[x]=x;
        } FOR(i,1,tot) if (son[id[i]]) top[son[id[i]]]=top[id[i]];
        rFOR(i,1,tot) ladder[top[id[i]]].push_back(id[i]);
        FOR(i,2,tot) {
            int x=id[i];
            if (top[x]==x) {
                for (int y=fa[x][0],c=len[x]; y&&c;
y=fa[y][0],c--)
                    ladder[x].push_back(y);
            }
        } upp[0]=-1;
        FOR(i,1,tot) upp[i]=upp[i-1]+(i==(i&-i));
    }
    int prev(int x,int k) {
        if (!k) return x;
        if (dep[x]<=k) return 0;
        x=fa[x][upp[k]]; k-=1<<upp[k];
        k-=dep[x]-dep[top[x]]; x=top[x];
        return ladder[x][len[x]+k];
    }
}
using namespace ladder;
```

最大团

```
int n,ans;
int edge[maxn][maxn],cnt[maxn],vis[maxn];//vis:元素
bool dfs(int u,int pos) {
    int i,j;
    FOR(i,u+1,n) {
        if (cnt[i]+pos<=ans) return 0;
        if (edge[u][i]) {
            REP(j,pos) if (!edge[i][vis[j]]) break;
            if (j==pos) {
                vis[pos]=i;
                if (dfs(i,pos+1)) return 1;
            }
        }
    }
    if (pos>ans) {ans=pos;return 1;}
    return 0;
    }
    int maxclique() {
        int i; ans=-1;
        rFOR(i,1,n) {
            vis[0]=i;
            dfs(i,1);
            cnt[i]=ans;
        } return ans;
    }
```

最小树形图(mlogn)

//不定根:新加一个节点，向所有点加一条 INF 的边，最后减一下即可

//主要思路:缩点

//输出路径思路:缩完点记录边,然后新建边记录等价关系

nm 的做法:

```
namespace O {
    const int maxn=1e5+7;
    const int maxm=1e5+7;
    struct Edge {//id,pre!=0;uid:替换
        int u,v,len;//id->usedID(new),用于新建边
        Edge(int _u=0,int _v=0,int _len=0):
            u(_u),v(_v),len(_len) {}
    } edge[maxm]; int etot;
    void init() {etot=0;}
    void addedge(int u,int v,int len) {
        edge[++etot]=Edge(u,v,len);
    }
    struct _info {//pre 为 preuid
        int pre,len,eid;//position(id) and length
        void init(int _pre,int _len,int _eid){
            pre=_pre; len=_len; eid=_eid;
        }
    } Info[maxn*2];
    //id:circle_id(father);top:tree_anc
    inline int getfa(int fa[],int x) {
        if (fa[x]==x) return x;
        return fa[x]=getfa(fa,fa[x]);
    } int id[maxn*2],top[maxn*2];//并查集;
    int idfa[maxn*2],useid[maxn*2];//changes_fa
    int used[maxm];//output; 记录 edge
    int solve(int root,int n) {
        int ret=0,i,lastnid=n;
        FOR(i,1,n) id[i]=top[i]=i,Info[i].len=INF;
        FOR(i,1,etot) { //initialize
            Edge &e=edge[i];
            if (e.u!=e.v&&e.len<Info[e.v].len)
                Info[e.v].init(e.u,e.len,i);
        }
        FOR(i,1,lastnid) {
            if (i==root) continue;
            if (Info[i].len==INF) return -1;
            _info &info=Info[i];
            int f=getfa(top,info.pre);
            //choose; 之后再更新
            useid[i]++; ret+=info.len; idfa[i]=i;
            if (f==i) {//circle
                int k; ++lastnid; Info[lastnid].len=INF;
                top[lastnid]=id[lastnid]=lastnid;
                for (int
x=getfa(id,info.pre); ;x=getfa(id,Info[x].pre)) {
                    FOR(k,1,etot) {
                        Edge &e=edge[k];
                        if (k==info.eid) e.len=INF;
                        if (e.len==INF) continue;//removed
                        if (getfa(id,e.v)==x) //must_ok
                            e.len-=Info[x].len;
                    }
                    //use and delete
                    id[x]=top[x]=idfa[x]=lastnid;//缩环
                    if (x==i) break;
                } //update edges
                FOR(k,1,etot) {
                    Edge &e=edge[k];
                    if (e.len==INF) continue;//removed
                    if (getfa(id,e.v)==lastnid) {//must_ok
                        if
(getfa(id,e.u)!=lastnid&&e.len<Info[lastnid].len)
                            Info[lastnid].init(e.u,e.len,k);//直
接这样应该 ok?
                    }
                }
            } else top[getfa(id,i)]=info.pre;//getfa=getid
        }
        rFOR(i,1,lastnid) if (useid[i]) {//remove_to_top
            static int ids[maxn];
            int x=0,k;
            for (int k=edge[Info[i].eid].v;k!=i;k=idfa[k])
ids[x++]=k;
```

```
            REP(k,x) idfa[ids[k]]=i,useid[ids[k]]=0;
            used[Info[i].eid]=1;
        }
        return ret;
    }
}
```

mlogn 的做法：

```
namespace heap {
    const int maxn=1e5+7;
    struct node {
        int l,r,len;
        int u,v,val,lz;
        node(int _u=0,int _v=0,int _val=0):
            u(_u),v(_v),val(_val) {l=r=len=0; lz=0;}
    } T[maxn]; int tot;
    //不能直接 swap x 和儿子，否则可能不满足堆性质
    void update(int x,int val) {
        T[x].lz+=val; T[x].val-=val;
    }
    void pushdown(int x) {
        if (T[x].lz) {
            if (T[x].l) update(T[x].l,T[x].lz);
            if (T[x].r) update(T[x].r,T[x].lz);
            T[x].lz=0;
        }
    }
    int merge(int x,int y) {
        if (!x||!y) return x|y;
        pushdown(x); pushdown(y);
        if (T[x].val>T[y].val) swap(x,y);
        T[x].r=merge(T[x].r,y);
        if (T[T[x].l].len<T[T[x].r].len) swap(T[x].l,T[x].r);
        T[x].len=T[T[x].r].len+1;
        return x;
    }
    int pop(int x) {
        pushdown(x);
        return merge(T[x].l,T[x].r);
    }
}
namespace O {
    const int maxn=1e5+7;
    const int maxm=1e5+7;
    int Root[maxm],etot;
    void init() {etot=0;}
    void addedge(int u,int v,int len) {
        heap::T[++etot]=heap::node(u,v,len);
    }
    struct _info {//pre 为 preuid;maxtot=maxm
        int pre,len,eid;//position(id) and length
        void init(int _pre,int _len,int _eid) {
            pre=_pre; len=_len; eid=_eid;
        }
    } Info[maxm];
    //id:circle_id(father);top:tree_anc
    inline int getfa(int fa[],int x) {
        if (fa[x]==x) return x;
        return fa[x]=getfa(fa,fa[x]);
    } int id[maxm],top[maxm];//并查集;
    bool getTopValue(int i) {
        while (Root[i]) {
            heap::node &e=heap::T[Root[i]];
            if (getfa(id,e.u)==getfa(id,e.v)) {
                Root[i]=heap::pop(Root[i]);
            } else {
                Info[i].init(e.u,e.val,Root[i]);
                Root[i]=heap::pop(Root[i]);
                // printf("%d: %d;
pos=%d\n",e.v,e.val,Root[i]);
                return 1;
            }
        } return 0;
    }
    int idfa[maxm],useid[maxm];//changes_fa
    int used[maxm];//output; 记录 edge
    int solve(int root,int n) {
        int ret=0,i,lastnid=n;
        FOR(i,1,n) id[i]=top[i]=i,Info[i].len=INF;
        FOR(i,1,etot) { //initialize
            heap::node &e=heap::T[i];
            Root[e.v]=heap::merge(Root[e.v],i);
        }
        FOR(i,1,n) if (i!=root&&!getTopValue(i)) return -1;
        // puts("ok1");
        FOR(i,1,lastnid) {
            if (i==root) continue;
            _info &info=Info[i];
            int f=getfa(top,info.pre);
            //choose; 之后再更新
            useid[i]++; ret+=info.len; idfa[i]=i;
            if (f==i) {//circle
                ++lastnid; Info[lastnid].len=INF;
                top[lastnid]=id[lastnid]=lastnid;
                for (int x=getfa(id,info.pre); ;
x=getfa(id,Info[x].pre)) {
                    heap::update(Root[x],Info[x].len);

Root[lastnid]=heap::merge(Root[lastnid],Root[x]);
                    id[x]=top[x]=idfa[x]=lastnid;//缩环
                    if (x==i) break;
                } //update edges
                if (!getTopValue(lastnid)) return -1;
            } else top[getfa(id,i)]=info.pre;//getfa=getid
        }
        rFOR(i,1,lastnid) if (useid[i]) {//remove_to_top
            static int ids[maxn];
            int x=0,k;
            for (int k=heap::T[Info[i].eid].v; k!=i; k=idfa[k])
ids[x++]=k;
            REP(k,x) idfa[ids[k]]=i,useid[ids[k]]=0;
            used[Info[i].eid]=1;
        }
        return ret;
    }
    //debug
    vector<pair<int,int> > check_edge[maxn];
    bool vis[maxn];
    void addcheck_edge_check(int u,int v,int i) {
        check_edge[u].push_back(make_pair(v,i));
    }
    void bfs(int x,int n) {//x=root
        queue<int> Q;
        Q.push(x); vis[x]=1;
        while (Q.size()) {
            int x=Q.front(); Q.pop();
            for (auto e:check_edge[x]) {
                int i=e.second;
                if (vis[e.first]) continue;
                if (!used[i]) continue;
                Q.push(e.first); vis[e.first]=1;
            }
        } int i;
        FOR(i,1,n) {
            if (!vis[i]) debug("no! %d\n",i);
        }
        }
}
int ini[maxn];
int n,m,i;
int u,v,w;
int main() {
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);
    scanf("%d%d",&n,&m);
    O::init();
    FOR(i,1,m) {
        if (scanf("%d%d%d",&u,&v,&w)!=3) {
            puts("input not right");
            return 0;
        };
        ini[i]=w;
        O::addedge(u,v,w);
        O::addcheck_edge_check(u,v,i);
    }
    int ans=O::solve(1,n);
    printf("%d\n",ans);
    if (ans!=-1) {
        O::bfs(1,n);
        FOR(i,1,m) if (ini[i]&&O::used[i])
            printf("%d ",i),ans--;
        if (ans) printf("\nnotok: %d\n",ans);
    }
}
```

一般图最大匹配 带花树

```
//缩奇环
int n,m;
vector<int> edge[maxn];
bool inQueue[maxn];
int belong[maxn];
int getbelong(int x) {
    if (belong[x]==x) return x;
    return belong[x]=getbelong(belong[x]);
}
int match[maxn],nxt[maxn],mark[maxn],vis[maxn];
int cnt;
queue<int> Q;
int used[maxn];
int lca(int u,int v) {
    cnt++;
    while (1) {
        u=getbelong(u);
        if (vis[u]==cnt) return u;
        vis[u]=cnt;
        u=nxt[match[u]];
        if (v) swap(u,v);
    }
}
void merge(int u,int p) {
    while (u!=p) {
        int mu=match[u],v=nxt[mu];
        if (getbelong(v)!=p) nxt[v]=mu;
        if (mark[mu]==2) mark[mu]=1,Q.push(mu);
        if (mark[v]==2) mark[v]=1,Q.push(v);
        int x,y;
        x=getbelong(u),y=getbelong(mu);
        if (x!=y) belong[x]=y;
        x=getbelong(mu),y=getbelong(v);
        if (x!=y) belong[x]=y;
        u=v;
    }
}
void solve(int s) { //增广
    int i;
    FOR(i,1,n) belong[i]=i,mark[i]=nxt[i]=0;
    while (Q.size()) Q.pop();
    Q.push(s);
    while (Q.size()) {
        if (match[s]) return;
        int u=Q.front();
        Q.pop();
        for (int v:edge[u]) {
            if (match[u]==v) continue;
            if (getbelong(u)==getbelong(v)) continue;
            if (mark[v]==2) continue; //T 型点
            if (mark[v]==1) { //S 型点,缩点
                int p=lca(u,v);
                if (getbelong(u)!=p) nxt[u]=v;
                if (getbelong(v)!=p) nxt[v]=u;
                merge(u,p);
                merge(v,p);
            } else if (!match[v]) { //增广
                nxt[v]=u;
                for (int x=v; x;) {
                    int y=nxt[x],xx=match[y];
                    match[x]=y;
                    match[y]=x;
                    x=xx;
                }
                break;
            } else {
                nxt[v]=u;
                mark[match[v]]=1;
                Q.push(match[v]);
                mark[v]=2;
            }
        }
    }
}
bool E[maxn][maxn];
int ans;
int main() {
    scanf("%d%d",&n,&m);
    int i;
    while (m--) {
        int u,v;
        scanf("%d%d",&u,&v);
        if (u!=v&&!E[u][v]) {
            edge[u].push_back(v);
            edge[v].push_back(u);
            E[u][v]=E[v][u]=1;
        }
    }
    memset(match,0,sizeof(match));
    FOR(i,1,n) if (!match[i]) solve(i);
    FOR(i,1,n) if (match[i]) ans++;
    ans/=2;
    printf("%d\n",ans);
    FOR(i,1,n) printf("%d ",match[i]);
}
```

树分块 高度分块

```
//题意: 给两颗树, 树上有边
//问你: T1 的 1->i 和 T2 的 1->i 的所有边加入计算后有多少联通块
//做法: 先把两棵树分成 sqrt 个块, 将 query 放到两个树的块上
//然后直接从上面转移+回滚, 最多 sqrt 的 length
struct Changes {
    int x,y,ini;
```

```
        Changes(int _x=0,int _y=0,int _ini=0):x(_x),y(_y),ini(_ini)
{};
    } changes[maxn*50]; int top;
    vector<int> E[maxn],V1,V2;
    int fa1[maxn],fa2[maxn],SIZE;
    int id1[maxn],id2[maxn];
    int last[507][507];
    vector<Changes> queries[507][507];
    void dfs1(int x,int f,int dep) {
        fa1[x]=f; id1[x]=-1;
        if (!(dep%SIZE)) id1[x]=V1.size(),V1.push_back(x);
        for (int v:E[x]) if (v!=f) dfs1(v,x,dep+1);
    } void dfs2(int x,int f,int dep) {
        fa2[x]=f; id2[x]=-1;
        if (!(dep%SIZE)) id2[x]=V2.size(),V2.push_back(x);
        for (int v:E[x]) if (v!=f) dfs2(v,x,dep+1);
    }
    int ux[maxn],uy[maxn],vx[maxn],vy[maxn];
    int fa[maxn],sz[maxn],nowans;
    void merge_(int x,int y) {
        while (x!=fa[x]) x=fa[x];
        while (y!=fa[y]) y=fa[y];
        if (x==y) return;
        if (sz[x]>sz[y]) swap(x,y);
        changes[++top]=Changes(x,y);
        fa[x]=y; nowans--; sz[y]+=sz[x];
        debug("merge_OK (%d %d)-%d
ans=%d\n",x,y,top,nowans);
    }
    void revert(int x,int y) {
        debug("revert to %d %d\n",x,y);
        while (top>last[id1[x]][id2[y]]) {
            auto now=changes[top--]; nowans++;
            fa[now.x]=now.x; fa[now.y]=now.y;
            sz[now.y]-=sz[now.x];
            debug("revert_OK (%d %d)-%d+1
ans=%d\n",now.x,now.y,top,nowans);
        }
    }
    void commit(int x,int y) {
        int tx=x,ty=y;
        while (id1[tx]==-1) tx=fa1[tx];
        while (id2[ty]==-1) ty=fa2[ty];
//      if (tx!=lastx||ty!=lasty)
        revert(tx,ty);
        while (x!=tx) merge_(ux[x],vx[x]),x=fa1[x];
        while (y!=ty) merge_(uy[y],vy[y]),y=fa2[y];
    }
    int ans[maxn];
    int main() {
        int T;
        int i,j,k;
        scanf("%d",&T);
        while (T--) {
            int n,m;
            scanf("%d%d",&n,&m);
            FOR(i,1,n) scanf("%d%d",&ux[i],&vx[i]);
            SIZE=sqrt(n)*1.1; if (SIZE==0) SIZE=1;
            FOR(i,1,n-1) {
                int u,v;
                scanf("%d%d",&u,&v);
                E[u].push_back(v);
                E[v].push_back(u);
            } V1.clear(); dfs1(1,0,0);
            FOR(i,1,n) E[i].clear();
            FOR(i,1,n) scanf("%d%d",&uy[i],&vy[i]);
            FOR(i,1,n-1) {
                int u,v;
                scanf("%d%d",&u,&v);
                E[u].push_back(v);
                E[v].push_back(u);
            } V2.clear(); dfs2(1,0,0);
            FOR(i,1,n) E[i].clear();
//          FOR(i,1,n) debug("%d %d\n",fa1[i],id1[i]);
//          FOR(i,1,n) debug("%d %d\n",fa2[i],id2[i]);
            FOR(i,1,n) {
                int u,v,x,y;
                u=v=i; x=u; y=v;
                while (id1[x]==-1) x=fa1[x];
                while (id2[y]==-1) y=fa2[y];
                queries[id1[x]][id2[y]].push_back(Changes(u,v,i));
            } deputs("okay");
            FOR(i,1,m) fa[i]=i,sz[i]=1;
            top=0; nowans=m;
            for (int x:V1) {
                for (int y:V2) {
                    if (x==1&&y==1) {
                        merge_(ux[x],vx[x]);
                        merge_(uy[y],vy[y]);
                    } else if (y==1) {
```

```
                    commit(fa1[x],y);
                    merge_(ux[x],vx[x]);
                } else {
                    commit(x,fa2[y]);
                    merge_(uy[y],vy[y]);
                } last[id1[x]][id2[y]]=top;
                for (auto now:queries[id1[x]][id2[y]]) {
                    debug("query %d %d\n",now.x,now.y);
                    commit(now.x,now.y);
                    ans[now.ini]=nowans;
                } queries[id1[x]][id2[y]].clear();
            }
        }
        FOR(i,1,n) printf("%d\n",ans[i]);
    }
    return 0;
}
```

树哈希

```
// 题意: A 树有多少个和 B 树同构的子树(B 总共 12 个节点)
// 做法: 对 B 树进行哈希, 然后构造转移方案并转移
namespace tree_hash {
    const int maxk=12;
    vector<int> baseedge[1<<maxk|7];
    void init(int n) {
        int i,j;
        REP(i,(1<<n)) REP(j,n) if ((i>>j)&1)
            baseedge[i].push_back(j);
    }
    typedef unsigned int type;
    struct tree {
        int e[maxk];//baseedge
        void init(int n) {memset(e,0,n*sizeof(int));}
        type dfs(int x,int fa) {//encode
            // printf("dfs: %d %d\n",x,fa);
            static int sz[maxk];
            type ret=0; sz[x]=1;
            vector<pair<type,int> > tmp;//count
            for (int v:baseedge[e[x]]) if (v!=fa) {
                    type val=dfs(v,x)<<1|1; sz[x]+=sz[v];
                    tmp.push_back(make_pair(val,sz[v]*2));
                } sort(tmp.begin(), tmp.end());
            reverse(tmp.begin(), tmp.end());
            for (pair<type,int>v:tmp)
                ret=(ret<<v.second)|v.first;
            return ret;
        }
        type unrooted(int n) {
            vector<type> tmp; int i;
            REP(i,n) tmp.push_back(dfs(i,i));
            sort(tmp.begin(), tmp.end());
            return tmp[0];//字典序 minimize
        }
    };
    set<type> S;//hash_answer
    typedef pair<tree,type> ptt;
    vector<pair<tree,type> > Trees[13];//size
    tree merge(const tree &x,const tree &y,int sizex,int sizey)
{
        tree ret; int i,j;
        ret.init(sizex+sizey);
        REP(i,sizex) REP(j,sizex) if ((x.e[i]>>j)&1) {
            ret.e[i]|=1ull<<j;
            ret.e[j]|=1ull<<i;
        }
        REP(i,sizey) REP(j,sizey) if ((y.e[i]>>j)&1) {
            ret.e[i+sizex]|=1ull<<(j+sizex);
            ret.e[j+sizex]|=1ull<<(i+sizex);
        }
        ret.e[0]|=1ull<<sizex;
        ret.e[sizex]|=1ull<<0;
        return ret;
    }
    int tot;
    short id[1<<(maxk*2)];//last=0 所以可以去掉
    typedef pair<type,int> pti;
    vector<pti> edges[8007];//这里是个暴力, 在外边再搞一次比
较好
    void getall(int n) {
        tree ini; ini.init(1);//ini.dfs=0
        Trees[1].push_back(make_pair(ini,0));
        int _,o; tot=0;
        S.insert(0); id[0]=++tot;
        FOR(_,1,n-1) {//size
            for (ptt tmp:Trees[_]) {
                int i; tree &ori=tmp.first;
                REP(i,_) {
                    tree nxt=ori;
                    nxt.e[_]=1ull<<i;
```

```
                        nxt.e[i]|=1ull<<_;
                        type v=nxt.dfs(0,0);
                        if (S.count(v)) continue;
                        Trees[_+1].push_back(make_pair(nxt,v));
                        S.insert(v); id[v]=++tot;
                        // assert(v<(1<<(maxk*2-1)));
                    }
                }
            }
            // int all=0;
            // FOR(_,1,n) {
            //   printf("%d: %d\n",_,(int)Trees[_].size());
            //   all+=Trees[_].size();
            //   // for (tree ori:Trees[_]) pr2(ori.dfs(0,0),_*2-2);
puts("");
            // } printf("size_all=%d\n",all);
            // printf("tot=%d\n",tot);
            FOR(_,1,n) FOR(o,1,n-_){
                for (ptt tmpx:Trees[_]) {
                    tree &x=tmpx.first; type ox=tmpx.second;
                    for (ptt tmpy:Trees[o]) {
                        tree &y=tmpy.first; type
oy=tmpy.second;
                        type o_nxt=merge(x,y,_,o).dfs(0,0);

edges[id[ox]].push_back(make_pair(id[oy],id[o_nxt]));
                    }
                }
            }
        }
    }
    vector<int> edge[2007];
    int value[2007][7900],tmp[7900],all[7900];
    void dfs(int x,int fa) {
        int i; value[x][1]=1;//base
        for (int v:edge[x]) if (v!=fa){
            dfs(v,x);
            FOR(i,1,tree_hash::tot) tmp[i]=0;
            FOR(i,1,tree_hash::tot) if (value[x][i])
                for (tree_hash::pti k:tree_hash::edges[i])

add_(tmp[k.second],(ll)value[x][i]*value[v][k.first]%M);
            FOR(i,1,tree_hash::tot) add_(value[x][i],tmp[i]);
        }
        // printf("dfs: %d %d\n",x,fa);
        // FOR(i,1,tree_hash::tot) if (value[x][i])
printf("%d: %d\n",i,value[x][i]);
        FOR(i,1,tree_hash::tot) add_(all[i],value[x][i]);
    }
    int main() {
        tree_hash::init(12);
        tree_hash::getall(12);
        int n,i,q;
        scanf("%d",&n);
        FOR(i,1,n-1) {
            int u,v; scanf("%d%d",&u,&v);
            edge[u].push_back(v);
            edge[v].push_back(u);
        } dfs(1,0);
        scanf("%d",&q);
        while (q--) {
            int m;
            tree_hash::tree tmp; tmp.init(12);
            scanf("%d",&m);
            vector<tree_hash::type> V;
            FOR(i,1,m-1) {
                int u,v;
                scanf("%d%d",&u,&v);
                u--; v--;
                tmp.e[u]|=1ull<<v;
                tmp.e[v]|=1ull<<u;
            } REP(i,m) V.push_back(tree_hash::id[tmp.dfs(i,i)]);
            sort(V.begin(), V.end());
            V.erase(unique(V.begin(), V.end()),V.end());
            int ans=0;
            for (int v:V) add_(ans,all[v]);
            printf("%d\n",ans);
        }
    }
```

数学相关

牛顿迭代 开根

```
C = int(raw_input())
for i in range(0, C):
    n = int(raw_input())
    if n < 2 :
        print n
        continue
    m = 2
    tmpn, len = n, 0
    while tmpn > 0:
        tmpn /= 10
        len += 1
    base, digit, cur = 300, len / m, len % m
    while (cur + m <= base) and (digit > 0):
        cur += m
        digit -= 1
    div = 10 ** (digit * m)
    tmpn = n / div
    x = int(float(tmpn) ** (1.0 / m))
    x *= (10 ** digit)
    while True:
        x0 = x
        x = x + x * (n - x ** m) / (n * m)
        if x == x0: break
    while (x + 1) ** m <= n:
        x = x + 1
    print x % 2
```

逆元, kummer 等基础

```
// 计数题的时候,"选择"和"方案"分开算是一个不错的选择~
// 注意 n>M 时要用 lucas!
LL inv[1000002];//inverse
LL fac[1000002];//Factorial
// 求出的是 ax+by=1 的解(a,b 正负不限,而且挺小的);
// d(gcd)==1 时存在逆元;(d!=1)&&(num|d)时,num*a/d 可认为
逆元
// (x+p)%p 为逆元
// DP:C[i][j]=(C[i-1][j-1]+C[i][j-1])%M
void exgcd(LL a,LL b,LL &d,LL &x,LL &y) {
    if (!b) {d=a; x=1; y=0;}
    else {exgcd(b,a%b,d,y,x); y-=a/b*x;}
}
// 前面那个线性求逆元的 log 版 2333
int getinv(int n) {
    if (n==1) return 1;
    return (M-M/n)*(getinv(M%n))%M;
}
LL C(int n,int m) {
    return fac[n]*inv[m]%M*inv[n-m]%M;
}
//Lucas 扩展: Kummer 定理:
//C(n,k)中的 p 的幂次的为 p 进制下 n-k 借位次数
//e.g.求 C(n,0)...C(n,n)的 lcm%(1e9+7)
//做法:考虑每个素因子,n 转化为 p 进制后,除了最后的为 p-1 的都可
以借位
//ans=pow(p,k)的乘积
LL lucas(LL n,LL m) { //注意 MOD 不能太大=_=! Mlogn
    return m==0?1:1ll*C(n%M,m%M)*lucas(n/M,m/M)%M;
}
```

BSGS

BSGS：a^x = b (mod p)
做法：假设 m=sqrt(p)+1; x=i*m-j(0<i<j)
枚举 i 和 j，我们得到了一个 sqrt(p)的做法

Pell 方程

$$x^2 - D * y^2 = n$$

打表求出第一项,然后下面的项可以线性递推

$$x_k + \sqrt{D}y_k = \left(x_1 + \sqrt{D}\, y_1\right)^k$$

$$x_{n+1} = x_0\, x_n + Dy_0\, y_n$$

博弈：NIM,SG

威佐夫博奕:
奇异局势:a=(b-a)*(sqrt(5)+1)/2;(lose)
NIM 博弈:k 堆石子，两人轮流每次从某一堆拿走一些

石子，问谁赢

做法:抽象(SG)->直接异或,sg=0(lose)

SG:选择的最多次数(连续,如 1,2,3...)

选择的最多次数,main 中为异或!=0

int sg[maxm+2];//打表~~~

这个是状态和剩余个数有关的

```
map<int,int> Hash;
int SG(int mask){
    if (Hash.count(mask)) return Hash[mask];
    set<int> mex;
    for (int i=0;i<maxm;++i){
        if (!((mask>>i)&1)) continue;//continue
        int tp=mask;
        for (int j=i;j<maxm;j+=i+1)//change
            if ((mask>>j)&1) tp^=1<<j;
        mex.insert(SG(tp));//dfs
    }
    int ret=0;
    for (;mex.count(ret);++ret);
    return Hash[mask]=ret;
}
```

这个是状态和剩余个数无关的

```
map<LL,int> Hash[62];
int SG(int x,LL mask){
    if (Hash[x].count(mask)) return Hash[x][mask];
    set<int> mex;
    for (int i=1;i<=x;++i){
        if ((mask>>(i-1))&1) continue;//continue
        int tp=mask;
        tp^=1<<(i-1);//change
        mex.insert(SG(x-i,tp));//dfs
    }
    int ret=0;
    for (;mex.count(ret);++ret);
    return Hash[x][mask]=ret;
}
```

Exgcd

```
//ax+by%x=y
int n,m;
int i,j,k;
void exgcd(LL a,LL b,LL &d,LL &x,LL &y){//d==0 时存在逆元
//(x+p)%p 为逆元
    if (!b) {d=a;x=1;y=0;}
    else {exgcd(b,a%b,d,y,x);y-=a/b*x;}
}
bool check(LL a,LL b,LL x){
    LL A,B,d;exgcd(a,b,d,A,B);
    A*=x;B*=x;
    LL T=A/b+B/a;
    A%=b;B%=a;
    if (A<0) A+=b,T--;
    if (B<0) B+=a,T--;
    return T>=0;
}
int solve(){
    int a,b,x,y;
    scanf("%lld%lld%lld%lld",&a,&b,&x,&y);
    int g=gcd(a,b);
    if (x%g||y%g) return 0*puts("NO");
    x/=g;y/=g;a/=g;b/=g;
    if (!(x%a)&&!(y%b)) return 0*puts("YES");
    if (!(y%a)&&!(x%b)) return 0*puts("YES");
    if (!(x%(a*b))&&check(a,b,y)) return 0*puts("YES");
    if (!(y%(a*b))&&check(a,b,x)) return 0*puts("YES");
    return 0*puts("NO");
}
```

K 次方和，伯努利数

```
//sum{pow(i,k)}(1->n)
ll B[maxn],pw[maxn];
ll A[maxn];
ll INV[10007];
LL inv[10002];//inverse
LL fac[10002];//Factorial
LL C(int n,int m) {
    return fac[n]*inv[m]%M*inv[n-m]%M;
} ll SUM_N_K(int n,int k) {
    ll pw=1,now=0; int i;
    FOR(i,1,k+1) {
        pw=pw*(n+1)%M;
        now+=INV[k+1]*C(k+1,i)%M*B[k+1-i]%M*pw%M;
    } mod_M(now);
    return now;
}
void initialize() {
    int i,j;
    fac[0]=1;
```

```
    FOR(i,1,10000) fac[i]=i*fac[i-1]%M;
    inv[0]=inv[1]=1; INV[0]=INV[1]=1;
    FOR(i,2,10000) INV[i]=inv[i]=(M-M/i)*inv[M%i]%M;
    FOR(i,1,10000) inv[i]=inv[i]*inv[i-1]%M;// inv(n!)
    B[0]=1;
    FOR(i,1,2000) {
        FOR(j,0,i-1) B[i]-=INV[i+1]*C(i+1,j)%M*B[j]%M;
mod_M(B[i]);
    }
//    FOR(i,0,2000) printf("%lld ",B[i]);
}
```

求原根 二次三次剩余(无板子)

原根:存在:m=2,4,p^a,2*p^a,p 为奇质数,个数 phi(phi(p-1))

查找:假设是 g,从小枚举 g

phi(m)=p1^a1*p2^a2*...*pk^ak;

pow(g,phi(m)/pi)≡1 恒成立(m 质数则 phi=m-1)

性质:pow(g,i)%p 得到的答案两两不同

推论 1 若 d|(p-1),则 x^d≡1(mod p)恰有 d 个解

推论 2 若 p 为素数,d|(p-1),则阶为 d (pow(x,d)≡1)

的最小剩余(mod p)的个数为 phi(d)

二次剩余:x*x≡n(mod p)

1.小的(a=0|p=2)直接判断

2.pow(n,(p-1)/2)≡1 或-1(mod p)

pow(n,(p-1)/2)≡1 则有解

3.由于 1/2 的数字有二次剩余

令 w=a*a-n;且 pow(n,(p-1)/2)≡-1

struct A+B*sqrt(w):

pow(a+sqrt(w),p)=pow(a,p)+pow(w,(p-1)/2)*sqrt(w))

≡a-sqrt(w)

pow(a+sqrt(w),p+1)≡a*a-w≡n

pow(a+sqrt(w),(p+1)/2)即为答案

三次剩余:x*x*x≡n(mod p)

1.小的(a=0|p=2,3)直接判断

2.p≡-1(mod 3):x≡pow(a,(2*p-1)/3)

3.p≡1(mod 3):设 e 为三次单位根,e*e*e≡1(mod p)

pow(a,(p-1)/3)≡1(mod p)则有三次剩余

```
int p[maxn],tot;
bool mark[maxn];
bool isroot(int x,int p){
    if (!(x%p)||(x%p==1&&p!=2)) return 0;
    for (ll i=2;i*i<=p-1;i++) if ((p-1)%i==0)
        if (poww(x,(p-1)/i,p)==1||poww(x,i,p)==1) return 0;
    return 1;
}
int TaskA() {
    int i,x;
    scanf("%d%d",&n,&x);
    if (mark[n+1]) return 0*puts("-1");
    rFOR(i,2,x-1){
        if (!isroot(i,n+1)) continue;
        return 0*printf("%d\n",i);
    } return 0*puts("-1");
}
void initialize() {
    int i,j;
    FOR(i,2,5000001) {
        if (!mark[i]) p[tot++]=i;
        REP(j,tot) {
            if (i*p[j]>5000001) break;
            mark[i*p[j]]=1;
            if (i%p[j]==0) break;
        }
    }
}
```

常系数线性递推

M^2logn 的普通版本:

```
int ini[3007];
const int mod=998244352;
void mul(ll *a,ll *b,int k) {
    int i,j;
    static ll tmp[3007];
    REP(i,k+k) tmp[i]=0;
    REP(i,k) if (a[i]) REP(j,k)
        ((tmp[i+j]+=a[i]*b[j])>=INFF)&&(tmp[i+j]%=mod);
    rrep(i,k,k+k) if (tmp[i]){
        tmp[i]%=mod;
        REP(j,k)
((tmp[i-k+j]+=tmp[i]*ini[k-j])>=INFF)&&(tmp[i-k+j]%=mod);
    }
    REP(i,k) a[i]=tmp[i]%mod;
}
ll A[3007],B[3007];
void power(int k,ll n) {
    if (k!=1) A[1]=1; else A[0]=ini[1]; B[0]=1;
    for (ll x=n; x; x>>=1) {
        if (x&1) mul(B,A,k);
        mul(A,A,k);
    }
}
```

Mlognlogm 的 fft 版本:前面的 fft 板子后面有的

```
ll A[maxn],B[maxn];
ll C[maxn],D[maxn],E[maxn];
int main() {
    ll l,r;
    scanf("%d%lld%lld",&n,&l,&r); int i;
    FOR(i,1,n) scanf("%lld",&A[i]),A[i]=B[i]=(M-A[i]%M);
    A[0]=1; while (A[n]==0) n--; m=n;
    n+=r-l; n++;
    NTT::inverse(A,A,n*2);
    m++; B[0]=1; reverse(B,B+m);//no!
    C[1]=1; D[0]=1; int lC=2,lD=1;
    for (ll x=l; x; x>>=1) {
        if (x&1) {
            NTT::multiply(C,D,D,lC,lD); int l1;
            NTT::delivery(D,B,E,D,lC+lD-1,m,l1,lD);
        } NTT::multiply(C,C,C,lC,lC); int l1;
        NTT::delivery(C,B,E,C,lC+lC-1,m,l1,lC);
    } reverse(D,D+lD);
    NTT::multiply(D,A,E,lD,n);
    FOR(i,lD-1,r-l+lD-1) printf("%lld\n",E[i]);
    return 0;
}
```

多项式暴力求积分

```
// 题意: 给 n 个区间, 求和与 0 的距离差期望
// 做法: 考虑每一次, 都是个区间分段积分形式
// 所以直接考虑每一段, 2-pointer 求 2^n 项系数即可
const int maxk=20;
int inv[maxk];
inline void init() {
    int i; inv[0]=1;
    rep(i,1,maxk) inv[i]=powMM(i,M-2);
}
struct poly {
    int A[maxk],n;//base
    void init(int n) {memset(A,0,sizeof(A)); this->n=n;};
    poly(int n=0) {init(n);}
    int getvalue(int x) {
        int i; ll ret=0;
        rREP(i,n+1) ret=(ret*x+A[i])%M;
        return ret;
    }
    poly integral() {//积分; ret.A[0]需要自己算
        poly ret; int i; ret.n=n+1; ret.A[0]=0;
        REP(i,n+1) ret.A[i+1]=(ll)A[i]*inv[i+1]%M;
        return ret;
    }
    poly derivative(){//求导
        poly ret; int i; ret.n=n-1; ret.A[0]=0;
        REP(i,n) ret.A[i]=(ll)A[i+1]*(i+1)%M;
        return ret;
    }
    poly move(poly base[]) {
        poly ret; ret.init(n); int i,j;
        REP(i,n+1) REP(j,i+1) {
            add_(ret.A[j],(ll)A[i]*base[i].A[j]%M);
        } return ret;
    }
    poly mul(const poly &p) {
        poly ret; ret.init(n+p.n); int i,j;
        REP(i,n+1) REP(j,p.n+1) {
            add_(ret.A[i+j],(ll)A[i]*p.A[j]%M);
        } return ret;
    }
    poly del(const poly &p) {
        poly ret; ret.init(max(n,p.n)); int i;
        REP(i,n+1) add_(ret.A[i],A[i]);
        REP(i,p.n+1) add_(ret.A[i],M-p.A[i]);
        return ret;
    }
}; //(x-r)
int len[maxn];
typedef pair<int,poly> pip;// 从 first 往后一段区间内的
poly_value 是 second
int base=0,multi=1;
void getintegral(pip now[],pip nxt[],int n) {//得到一个连续的积
分,并从 0 开始算常数项
    int i;
    FOR(i,1,n) {
        int k=nxt[i].first=now[i].first;
        nxt[i].second=now[i].second.integral();
        nxt[i].second.A[0]=(nxt[i-1].second.getvalue(k)
-nxt[i].second.getvalue(k)+M*2)%M;
        }
    }
    void getbase(poly base[],int k,int n) {//(x-k)^n; 用于移动整个
区间
        int i; base[0].init(0);
        base[0].A[0]=1;//1
        poly mul; mul.init(1);
        mul.A[0]=(M-k%M)%M; mul.A[1]=1;
        FOR(i,1,n) base[i]=base[i-1].mul(mul);
    }
    pip now[1<<15|7],inter[1<<15|7],nxt[1<<15|7];
    poly multibase[maxk];
    int ans=0; int mbase=1;
    int main() {
        init();
        int n,i;
        scanf("%d",&n);
        FOR(i,1,n) {
            int l,r;
            scanf("%d%d",&l,&r);
            add_(ans,(l+r)*powMM(2ll,M-2)%M);//之后加两倍负数
即可
            r-=l; base-=l; len[i]=r;
            if (r) mbase=mbase*powMM(r,M-2)%M;
        }
        reverse(len+1,len+1+n);
        now[0].second.init(0);
        now[0].second.A[0]=0;
        now[1].first=0;
        now[1].second.init(0);
        now[1].second.A[0]=1;//integeal_ed
        int m=1;//Count
        FOR(i,1,n) {
            if (len[i]==0) continue;
            getintegral(now,inter,m);
            //开始积分
            getbase(multibase,len[i],n);
            int l=1,r=1,nxtm=0;
            while (l<=m||r<=m) {
                int pos=INF;
                if (l<=m) pos=min(pos,now[l].first+len[i]);
                if (r<=m) pos=min(pos,now[r].first);
                while (l<=m&&now[l].first+len[i]==pos) l++;//相同
pos 只有一次
                while (r<=m&&now[r].first==pos) r++;
                ++nxtm; nxt[nxtm].first=pos;

nxt[nxtm].second=inter[r-1].second.del(inter[l-1].second.move(
multibase));
            }
            swap(now,nxt); m=nxtm;
        } poly x; x.init(1); x.A[1]=1;
        getbase(multibase,-base,n);
        FOR(i,1,m) {
            now[i].first-=base;
            now[i].second=now[i].second.move(multibase);
            now[i].second=now[i].second.derivative();
            now[i].second=now[i].second.mul(x);
        } getintegral(now,inter,m);
        int last=0;
        FOR(i,1,m) if (inter[i].first<=0) last=i;

add_(ans,(2ll*(M-(ll)inter[last].second.getvalue(0)*mbase%M))%M)
;
        printf("%d\n",ans);
    }
```

五边形数定理

```
/*hdu4651
  题意：普通的整数拆分
  限制：1 <= n <= 1e5
  思路：五边形数定理
  Q(x)=\mul(1-x^k) = 1-x-x^2+x^5+x^7+...
  Q(x)=\sum_k(-1)^k x^(k*(3k-1)/2)
 */
#include <iostream>
#include <cstdio>
using namespace std;
#define LL __int64
const int N=100005;
const int MOD=1000000007;
LL dp[N],fi[N];
LL five(LL x){ return (3*x*x-x)/2; }
//五边形数
void wbxs(){
    dp[0]=1;
    int t=1000; //其实可以等于 sqrt(N)base
    for(int i=-t;i<=t;++i)
```

```
        fi[i+t]=five(i);      //Q
    for(int i=1;i<=10000;++i){
        int flag=1;
        for(int j=1;;++j){
            LL a=fi[j+t],b=fi[-j+t];
            if(a>i && b>i) break;
            if(a<=i) dp[i]=(dp[i]+dp[i-a]*flag+MOD)%MOD;
//p
            if(b<=i) dp[i]=(dp[i]+dp[i-b]*flag+MOD)%MOD;
            flag*=-1;
        }
    }
}
```

FFT

DFT 式子: x_k=\sum{x_i*wn[k*i]};

```
namespace FFT {
    const int maxn=1<<18|7;
    struct complex {
        double a,b;
        complex(double _a=.0,double _b=.0):a(_a),b(_b) {}
        complex operator+(const complex x)const {return
complex(a+x.a,b+x.b);}
        complex operator-(const complex x)const {return
complex(a-x.a,b-x.b);}
        complex operator*(const complex x)const {return
complex(a*x.a-b*x.b,a*x.b+b*x.a);}
    };
    complex wn[maxn];
    void initwn(int l) {
        static int len=0; int i;
        if (len==l) return; else len=l;
        REP(i,len) wn[i]=complex(cos(2*pi*i/l),sin(2*pi*i/l));
    }
    void fft(complex *A,int len,int inv) {
        int i,j,k; initwn(len);
        for (i=1,j=len/2; i<len-1; i++) {
            if (i<j) swap(A[i],A[j]);
            k=len/2;
            while (j>=k) j-=k,k/=2;
            if (j<k) j+=k;
        } for (i=2; i<=len; i<<=1) {
            for (j=0; j<len; j+=i) {
                for (k=j; k<(j+i/2); k++) {
                    complex a,b; a=A[k];
                    b=A[k+i/2]*wn[(ll)(k-j)*len/i];
                    A[k]=a+b; A[k+i/2]=a-b;
                }
            }
        } if (inv==-1) REP(i,len)
A[i]=complex(A[i].a/len,A[i].b/len);
    }
    inline complex conj(complex &A) {return
complex(A.a,-A.b);}
    void mul(int *A,int *B,int *ans,int len) { //ans=A*B
        static complex x1[maxn],x2[maxn]; int i;
        REP(i,len) x1[i]=complex(A[i],B[i]);
        fft(x1,len,1);
        REP(i,len) {//这个 k1, b1 就是前面的, 这就减掉了一半常
数
            int j=(len-i)&(len-1);
            complex
a=(conj(x1[i])+x1[j])*complex(0.5,0);//dft a
            complex
b=(conj(x1[i])-x1[j])*complex(0,0.5);//dft b
            x2[i]=a*b;
        } fft(x2,len,-1);
        REP(i,len) ans[i]=x2[i].a+0.5;
    }
}
```

多项式单 log 无穷背包(MTT)

```
    // 主要思路不是这个裸的乘法啥的啊!
    // from picks' blog
    // 对 G(F(x))=0 进行泰勒展开
    // G'(F_{t+1}(x))=G(F_t(x))+G'(F_t(x))/1*(F_{t+1}-F_t(x))^1+....
    // 后方的系数在 mod x^2^t+1 的意义下全是 0!(因为减的那里的系
数是 2^t)
    // F_{t+1}(x)=F_t(x)-G(F_t(x))/G'(F_t(x))
    // 所以手动求个导数即可!
    // 注意这个 G(F(t))就是满足的那个式子! 注意要有常数项(否则可以
全是 0 =_=!)
    // 三角函数需要利用虚数来做, e^{iF(x)}=cos(F(x))+isin(F(x))
    // exp(x): F_{t+1}(x)=F_t(x)-F_t(x)*((ln(F_t(x))-P(x))*F_t(x))
    // ln(x) : ln(F(x))=\int(积分) F'(x)/F(x)
    // 注意 F[0]要是 0, 因为求导的时候会去掉这个贡献, 积分回来
// 这个是无穷背包
namespace FFT {
    const int maxn=1<<18|7;
    struct complex {
        double a,b;
        complex(double _a=.0,double _b=.0):a(_a),b(_b) {}
        complex operator+(const complex x)const {return complex(a+x.a,b+x.b);}
        complex operator-(const complex x)const {return complex(a-x.a,b-x.b);}
        complex operator*(const complex x)const {return complex(a*x.a-b*x.b,a*x.b+b*x.a);}
    };
    complex wn[maxn];
    void initwn(int l) {
        static int len=0; int i;
        if (len==l) return; else len=l;
        REP(i,len) wn[i]=complex(cos(2*pi*i/l),sin(2*pi*i/l));
    }
    void fft(complex *A,int len,int inv) {
        int i,j,k; initwn(len);
        for (i=1,j=len/2; i<len-1; i++) {
            if (i<j) swap(A[i],A[j]);
            k=len/2;
            while (j>=k) j-=k,k/=2;
            if (j<k) j+=k;
        } for (i=2; i<=len; i<<=1) {
            for (j=0; j<len; j+=i) {
                for (k=j; k<(j+i/2); k++) {
                    complex a,b; a=A[k];
                    b=A[k+i/2]*wn[(ll)(k-j)*len/i];
                    A[k]=a+b; A[k+i/2]=a-b;
                }
            }
        } if (inv==-1) REP(i,len) A[i]=complex(A[i].a/len,A[i].b/len);
    }
    inline complex conj(complex &A) {return complex(A.a,-A.b);}
    void mul(int *A,int *B,int *ans,int len,int mod) { //ans=A*B
        static complex x1[maxn],x2[maxn];
        static complex x3[maxn],x4[maxn];
        static const int S=1<<15 ; int i;
        REP(i,len) x1[i]=complex(A[i]/S,A[i]%S);
        REP(i,len) x2[i]=complex(B[i]/S,B[i]%S);
        fft(x1,len,1); fft(x2,len,1);
        REP(i,len) {//这个 k1, b1 就是前面的, 这就减掉了一半常数
            int j=(len-i)&(len-1);
            complex k1=(conj(x1[i])+x1[j])*complex(0.5,0);//dft k1
            complex b1=(conj(x1[i])-x1[j])*complex(0,0.5);//dft b1
            complex k2=(conj(x2[i])+x2[j])*complex(0.5,0);//dft k2
            complex b2=(conj(x2[i])-x2[j])*complex(0,0.5);//dft b2
            x3[i]=k1*k2+k1*b2*complex(0,1);
            x4[i]=b1*k2+b1*b2*complex(0,1);
        } fft(x3,len,-1); fft(x4,len,-1);
        REP(i,len) {
            ll kk=x3[i].a+0.5,kb=x3[i].b+0.5;
            ll bk=x4[i].a+0.5,bb=x4[i].b+0.5;

            ans[i]=((kk%mod*S%mod+kb+bk)%mod*S%mod+bb)%mod;
        }
    }

    const ll Mod=19260817;
    // 下方的东西和 ntt 就根本无关, 这个模数是可以改的, 是多项式相关的东西
    // 也就是说, 这个模数完全可以取其他的, 然后高精度的 mtt 来求, 不过可能会 T 到死
    int eInv[maxn];
    void initinv(int l) {
        int i; eInv[0]=eInv[1]=1;
        rep(i,2,l) eInv[i]=(Mod-Mod/i)*eInv[Mod%i]%Mod;
    }
    void Ftof(int *A,int *B,int l) {//derivative 求导
        int i;
        FOR(i,1,l) B[i-1]=(ll)A[i]*i%Mod;
    }
    void ftoF(int *A,int *B,int l) {//integral 积分
        int i; // todo:get B[0], getinv
        rFOR(i,1,l) B[i]=(ll)A[i-1]*eInv[i]%Mod;
        B[0]=0;
    }
    void inv(int *A,int *B,int l) { //B=inv(A)
        static int C[maxn],D[maxn];
        B[0]=eInv[A[0]]; B[1]=0;
        for (int len=2; len<=l; len<<=1) {
            int i; fill(B+len,B+len+len,0);
            copy(A,A+len,C); fill(C+len,C+len+len,0);
            mul(C,B,D,len*2,Mod); fill(D+len,D+len+len,0);
            mul(D,B,D,len*2,Mod);
            REP(i,len) B[i]=(B[i]*2-D[i]+Mod)%Mod;
            fill(B+len,B+len+len,0);
        }
    }
    void ln(int *A,int *B,int l) {
        static int C[maxn];
        inv(A,B,l); Ftof(A,C,l);
        mul(B,C,B,l*2,Mod);
        ftoF(B,B,l);
    }
    void exp(int *A,int *B,int l) {
        static int C[maxn],i;
        B[0]=1; B[1]=0;
        for (int len=2; len<=l; len<<=1) {
            fill(B+len,B+len+len,0);
            ln(B,C,len); fill(C+len,C+len+len,0);
            REP(i,len) C[i]=(C[i]-A[i]+Mod)%Mod;
            mul(B,C,C,len*2,Mod);
            REP(i,len) B[i]=(B[i]-C[i]+Mod)%Mod;
        }
    }
    //这里是更高一层的东西
    static int A[maxn],B[maxn];
    void multiply(int *a,int *b,int *ans,int n,int m)
{//C=A*B(actual)
        int len=1,i;
        while (len<n+m-1) len<<=1;
        REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;
        REP(i,m) B[i]=b[i]; rep(i,m,len) B[i]=0;
        mul(A,B,ans,len,Mod);
    }
    void getexp(int *a,int *ans,int n) {
        int len=1,i;
        while (len<n) len<<=1;
        REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;
        exp(A,ans,len);
    }
    void solve(int *a,int *ans,int m) {
        static int A[maxn];
        int i,j;
        FOR(i,1,m) {//无穷背包
            int now=(ll)i*a[i]%Mod;
            for (j=i-1; j<=m; j+=i) A[j]=(now+A[j])%Mod;
        } ftoF(A,A,m);
        getexp(A,ans,m+1);
    }
}
int A[maxn],ans[maxn];
int main() {
    int i,k;
    FFT::initinv(maxn);
    scanf("%d%d",&n,&m);
    FOR(i,1,n) scanf("%d",&k),A[k]++;
    FFT::solve(A,ans,m);
    // FOR(i,1,m) printf("%d ",ans[i]);
    int Ans=0;
    FOR(i,1,m) add_(Ans,ans[i],FFT::Mod);
    printf("%d\n",Ans);
}
```

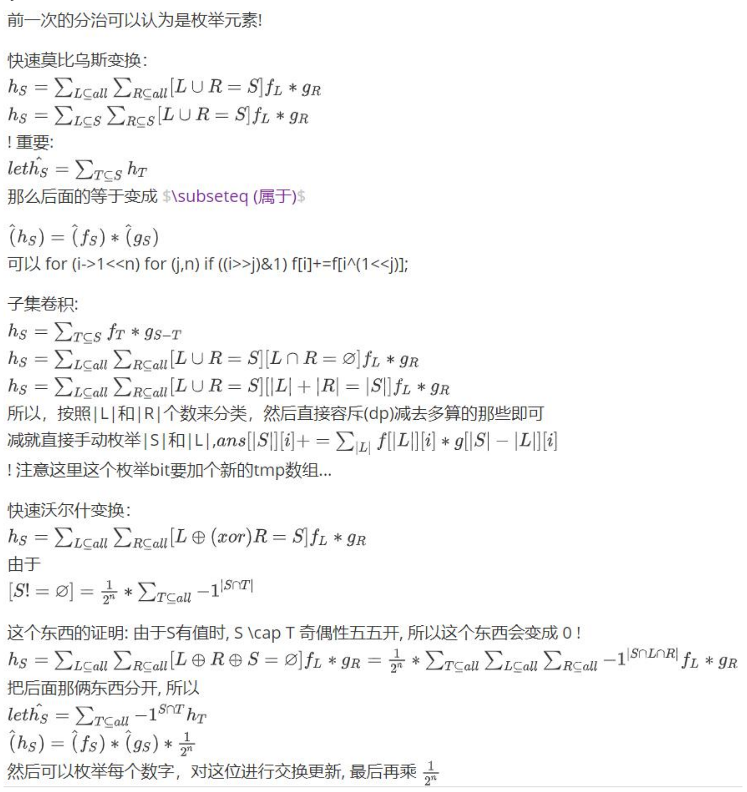
多项式开根求逆,除法取模(NTT)

```
// http://codeforces.com/contest/438/problem/E
// 题意: 问你有多少个二叉树点权从 c 中取, 而且权值和是 k
// 做法: 考虑多一个点, 所以 f[x]=sigma{f[k]*f[x-k-s],(s in c)}
// 所以 满足 F=F^2*C+1, 左边是生成函数
// 所以 F=[1-sqrt(1-4C)]/2C=1/(1+sqrt(1-4C))
// 当且仅当常数项有逆元, 可以多项式求逆
// 求逆:C*A≡1(mod x^n)
// B*A≡1(mod x^(n/2))
// (B*A-1)*(B*A-1)≡0(mod x^(n/2))
// B*B*A*A-2*A*B+1≡0(mod x^n)
// B*B*A-2*B+C≡0(mod x^n)
// C≡B*(2-A*B)(mod x^n)

// 求根:C*C≡A(mod x^n)
// B*B≡A(mod x^n/2)
// (B*B-A)*(B*B-A)≡0(mod x^n)
// B*B*B*B-2*C*C*B*B+C*C*C*C≡0(mod x^n)
// (B*B+C*C)*(B*B+C*C)≡4*C*C*B*B(mod x^n)
// B*B+A≡2*C*B(mod x^n)
// C=(B*B+A)/(2*B)
namespace NTT {
    const int maxn=1<<20|7;
    const ll MOD=998244353;
    const ll g=3;
    int wn[maxn],invwn[maxn];
    ll mul(ll x,ll y) {
        return x*y%MOD;
    }
    ll poww(ll a,ll b) {
        ll ret=1;
        for (; b; b>>=1ll,a=mul(a,a))
            if (b&1) ret=mul(ret,a);
        return ret;
    }
    void initwn(int l) {
        static int len=0;
        if (len==l) return; len=l;
        ll w=poww(g,(MOD-1)/len); int i;
        ll invw=poww(w,MOD-2); wn[0]=invwn[0]=1;
        rep(i,1,len) {
            wn[i]=mul(wn[i-1],w);
            invwn[i]=mul(invw,invwn[i-1]);
        }
    }
    void ntt(ll *A,int len,int inv) {
        int i,j,k; initwn(len);
        for (i=1,j=len/2; i<len-1; i++) {
            if (i<j) swap(A[i],A[j]);
            k=len/2;
            while (j>=k) j-=k,k/=2;
            if (j<k) j+=k;
        } for (i=2; i<=len; i<<=1) {
            for (j=0; j<len; j+=i) {
                for (k=j; k<(j+i/2); k++) {
                    ll a,b; a=A[k];
                    if (inv==-1)
b=mul(A[k+i/2],invwn[(ll)(k-j)*len/i]);
                    else b=mul(A[k+i/2],wn[(ll)(k-j)*len/i]);
                    A[k]=(a+b); (A[k]>=MOD)
&&(A[k]-=MOD);
                    A[k+i/2]=(a-b+MOD); (A[k+i/2]>=MOD)
&&(A[k+i/2]-=MOD);
                }
            }
        } if (inv==-1) {
            ll vn=poww(len,MOD-2);
            REP(i,len) A[i]=mul(A[i],vn);
        }
    }
    void mul(ll *A,ll *B,ll *C,int len) { //C=A*B
        int i;
        ntt(A,len,1); ntt(B,len,1);
        REP(i,len) C[i]=mul(A[i],B[i]);
        ntt(C,len,-1);
    }
    void inv(ll *A,ll *B,int l) { //B=inv(A)
        static ll C[maxn];
        B[0]=poww(A[0],MOD-2); B[1]=0;
        for (int len=2; len<=l; len<<=1) {
            int i; fill(B+len,B+len+len,0);
            copy(A,A+len,C); fill(C+len,C+len+len,0);
            ntt(C,len*2,1); ntt(B,len*2,1);
            REP(i,len*2)
B[i]=mul(B[i],(MOD+2-mul(C[i],B[i])));
            ntt(B,len*2,-1); fill(B+len,B+len+len,0);
        }
    }
    void sqrt(ll *A,ll *B,int l) { //B=sqrt(A)
        static ll C[maxn],_B[maxn];
        B[0]=1; B[1]=0;// 这里应该是个二次剩余
        for (int len=2; len<=l; len<<=1) {
            int i; ll inv2=poww(2,MOD-2);
            inv(B,_B,len); fill(B+len,B+len+len,0);
            copy(A,A+len,C); fill(C+len,C+len+len,0);
            ntt(C,len*2,1); ntt(_B,len*2,1); ntt(B,len*2,1);
            REP(i,len*2) B[i]=mul(inv2,B[i]+mul(C[i],_B[i]));
            ntt(B,len*2,-1); fill(B+len,B+len+len,0);
        }
    }
    static ll A[maxn],B[maxn];
    void multiply(ll *a,ll *b,ll *ans,int n,int m)
{//C=A*B(actual)
        int len=1,i;
        while (len<n+m-1) len<<=1;
        REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;
        REP(i,m) B[i]=b[i]; rep(i,m,len) B[i]=0;
        mul(A,B,ans,len);
    }
    void inverse(ll *a,ll *ans,int n){
        int len=1,i;
        while (len<n) len<<=1;
        REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;
        inv(A,ans,len);
    }
    void getsqrt(ll *a,ll *ans,int n){
        int len=1,i;
        while (len<n) len<<=1;
        REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;
        sqrt(A,ans,len);
    }
    void divide(ll *a,ll *b,ll *ans,int n,int m,int &l) {
        if (n<m) {l=1; ans[0]=0; return;}
        int len=1,i; l=n-m+1;
        while (len<n-m+1) len<<=1;
        REP(i,n) A[i]=a[i]; reverse(A,A+n); min_(n,l);
        REP(i,m) B[i]=b[i]; reverse(B,B+m); min_(m,l);
        rep(i,m,len) B[i]=0;
        inv(B,ans,len);
        multiply(A,ans,ans,len,n);
        reverse(ans,ans+l);
    }
    //ans1:答案; ans2:余数
    void delivery(ll *a,ll *b,ll *ans1,ll *ans2,int n,int m,int
&l1,int &l2) {
        divide(a,b,ans1,n,m,l1); l2=m-1;
        static ll tmp[maxn];
        multiply(b,ans1,tmp,m,l1);
        int i; REP(i,l2) ans2[i]=(a[i]-tmp[i]+M)%M;
    }
}
ll A[maxn],ans[maxn];
int main() {
    int i,k;
    scanf("%d%d",&n,&m);
    FOR(i,1,n) scanf("%d",&k),A[k]++;
    REP(i,m+1) A[i]=-4*A[i]; A[0]++;
    REP(i,m+1) mod_(A[i]);
    NTT::getsqrt(A,ans,m+1);
    add_(ans[0],1);
    NTT::inverse(ans,ans,m+1);
    FOR(i,1,m) mul_(ans[i],2);
    FOR(i,1,m) printf("%lld\n",ans[i]);
}
```

fwt，子集卷积

按数学理解:


前一次的分治可以认为是枚举元素!

快速莫比乌斯变换:
$h_S = \sum_{L\subseteq all} \sum_{R\subseteq all} [L \cup R = S] f_L * g_R$
$h_S = \sum_{L\subseteq S} \sum_{R\subseteq S} [L \cup R = S] f_L * g_R$
! 重要:
$let \hat{h_S} = \sum_{T\subseteq S} h_T$
那么后面的等于变成 $\subseteq (属于)$

$\hat{(}h_S) = \hat{(}f_S) * \hat{(}g_S)$
可以 for (i->1<<n) for (j,n) if ((i>>j)&1) f[i]+=f[i^(1<<j)];

子集卷积:
$h_S = \sum_{T\subseteq S} f_T * g_{S-T}$
$h_S = \sum_{L\subseteq all} \sum_{R\subseteq all} [L \cup R = S][L \cap R = \varnothing] f_L * g_R$
$h_S = \sum_{L\subseteq all} \sum_{R\subseteq all} [L \cup R = S][|L| + |R| = |S|] f_L * g_R$
所以，按照|L|和|R|个数来分类，然后直接容斥(dp)减去多算的那些即可
减就直接手动枚举|S|和|L|,$ans[|S|][i] += \sum_{|L|} f[|L|][i] * g[|S| - |L|][i]$
! 注意这里这个枚举bit要加个新的tmp数组...

快速沃尔什变换:
$h_S = \sum_{L\subseteq all} \sum_{R\subseteq all} [L \oplus (xor) R = S] f_L * g_R$
由于
$[S! = \varnothing] = \frac{1}{2^n} * \sum_{T\subseteq all} -1^{|S\cap T|}$

这个东西的证明: 由于S有值时, S \cap T 奇偶性五五开, 所以这个东西会变成 0 !
$h_S = \sum_{L\subseteq all} \sum_{R\subseteq all} [L \oplus R \oplus S = \varnothing] f_L * g_R = \frac{1}{2^n} * \sum_{T\subseteq all} \sum_{L\subseteq all} \sum_{R\subseteq all} -1^{|S\cap L\cap R|} f_L * g_R$
把后面那俩东西分开, 所以
$let \hat{h_S} = \sum_{T\subseteq all} -1^{S\cap T} h_T$
$\hat{(}h_S) = \hat{(}f_S) * \hat{(}g_S) * \frac{1}{2^n}$
然后可以枚举每个数字，对这位进行交换更新, 最后再乘 $\frac{1}{2^n}$

按位理解:

//or/and 的理解:这里的变换是利用 dp 时分治来压位(写成非递归形式)实现的，时间 nlogn

//进行组合可以将二元运算的东西都组合出来

//实际上 or 都没用

```
void fwt(LL *A,int len,int inv)//对拍对了
{
    int i,j,k;
    int div=powMM(2ll,M-2);
    for(i=2;i<=len;i<<=1){
        for(j=0;j<len;j+=i){
            for(k=j;k<j+i/2;k++){
                if (inv==1){
                    LL a=A[k],b=A[k+i/2];
                    A[k]=(a+b)%M;
                    A[k+i/2]=(a-b+M)%M;
                //xor:a[k]=x+y,a[k+i/2]=(x-y+mod)%mod;
                //and:a[k]=x+y;
                //or:a[k+i/2]=x+y;
                }else{
                    LL a=A[k],b=A[k+i/2];
                    A[k]=(a+b)*div%M;
                    A[k+i/2]=(a-b+M)%M*div%M;
                //xor:a[k]=(x+y)/2,a[k+i/2]=(x-y)/2;
                //and:a[k]=x-y;
                //or:a[k+i/2]=y-x;
                }
            }
        }
    }
}
```

子集和(高维前缀和，分治)

大概做法是按照每一位来分类，然后往下递归获得答案

就是，按照这一位是 1 和 0 分成几类往下递归

对子集分治和高次前缀和的理解:

子集分治:

考虑维护 f(x)=真正的值;

g(x)=子集中的贡献值(这个 g 可以是 f 进行 fft 得到的,也可以是与位有关的,较容易得到)

solve(x)的过程:

solve(left); 加 f 贡献到 g;

solve(right); 在 f=g 固定 right 结果; 若 g 需要更改则更改 g(如子集和就是普通的右边加左边)

高维前缀和:

高维前缀和的方式就是从小往大枚举每一位,然后一个一个维度做前缀和，从小往大 for value

然后对于每个 value,枚举前面的 cnt 次数,保存枚举的位数,分别前缀和

// f(x)=sum_(f(y)), f(x)在某些条件下=0

分治做法：

```
int f[1<<21|7],g[1<<21|7];
char str[1<<21|7];
void solve(int l,int r) {
    if (l+1==r) {
        if (!l) f[l]=1,g[l]=0;
        else if (str[l-1]=='+') f[l]=0;
        add_(g[l],f[l]);
        return;
    } int i,mid=(l+r)/2;
    solve(l,mid);
    REP(i,mid-l) add_(f[i+mid],g[i+l]);//left 的影响
    solve(mid,r);
    REP(i,mid-l) add_(g[i+mid],g[i+l]);
}
```

高维前缀和做法:

```
char str[1<<21|7];
int f[1<<21|7],g[1<<21|7][22];//sum_of_previous
int main() {
    int T,_; T=1;
    scanf("%d",&T);
    FOR(_,1,T){
        /*to solve the problem*/
        scanf("%s",str);
        int n=strlen(str),sta,i;
        memset(f,0,(n+2)*sizeof(int));
        int MAX=1;
        while ((1<<MAX)<=n+1) MAX++;
        f[0]=1; REP(i,MAX) g[0][i]=1;
        FOR(sta,1,n+1) {
            if(((n+1)&sta)==sta) {
                REP(i,MAX) {
                    g[sta][i]=i?g[sta][i-1]:0;
                    if ((sta>>i)&1)
add_(g[sta][i],g[sta^(1<<i)][i]);
                }
                if (str[sta-1]!='+') {
                    f[sta]=g[sta][MAX-1];//front
                    REP(i,MAX) add_(g[sta][i],f[sta]);
                }
            }
        }
        // REP(i,n+3) printf("%d ",f[i]); puts("<- f");
        // REP(i,n+3) printf("%d ",g[i][21]); puts("<- g");
        printf("%d %d\n",n+1,f[n+1]);
    }
}
```

子集卷积

//第一种做法: 按位考虑

//大概做法是按照每一位来分类，然后往下递归获得答案

就是，按照这一位是 1 和 0 分成几类往下递归

//http://acm.hdu.edu.cn/showproblem.php?pid=6057

//很容易卡 T...3^18 也许能过

//这个比 2^nlog^2(n=19)的慢了快 5 倍

//这种思路这种题都能用

//最好像 tls 那样推一推然后写成非递归，常数会减少到和 2^n*n^2 差不多

//真*子集卷积 by TLS(卡常)

```
const int maxn = 1 << 19 | 1, mod = 998244353, seed = 1526;
int n, all, bit[maxn], a[maxn], b[maxn], ans;
inline void mod_inc(int &x, int y) {
        (x += y) >= mod && (x -= mod);
    }
    int main() {
        while(scanf("%d", &n) == 1) {
            all = (1 << n) - 1;
            for(int i = 0; i <= all; ++i) scanf("%d", a + i);
            for(int i = 0; i <= all; ++i) scanf("%d", b + i);
            bit[0] = 1;
            for(int i = 1; i <= all; ++i) {
                bit[i] = bit[i >> 1] << (i & 1);
                a[i] = (LL)a[i] * bit[i] % mod;
            }
            ans = 0;
            for(int i = all; i >= 0; --i) {
                int msk = all ^ i, tim = 0;
                ULL cnt = 0;
                for(int j = msk; j; j = (j - 1) & msk) {
                    cnt += (ULL)a[j] * b[i | j];
                    (++tim) == 18 && (tim = 0, cnt %= mod);
                }
                cnt += (ULL)a[0] * b[i]; cnt %= mod;
                ans = ((LL)seed * ans + cnt) % mod;
            }
            printf("%d\n", ans);
        }
        return 0;
    }
```

//第二种做法: 数学方法

证明在上面小字

```
// C[k]=\sum_{i&j=k} A[i^j] B[i|j]
// let i'=i^j, j'=i|j, 这样的 i,j 对有 2^bit(i')个
// C[k]=\sum [j'-i'=k] [i' \subseteq j'] 2^i' * A[i'] * B[j']
// C[k]=\sum [i^j=k] [i&j=i] * 2^i * A[i] * B[j] //这里的意思就是 i|k=j, i+k=j
// C[k]=\sum [i^j=k] [bit(j)-bit(i)=bit(k)] 2^i * A[i] * B[j]
// ! 注意这里这个枚举 bit 要加个新的 tmp 数组...
int A[20][maxn],B[20][maxn],C[20][maxn];
int bit[maxn],pw1526[maxn],ans[maxn];
int main() {
    int k,i;
    scanf("%d",&m); n=1<<m;
    REP(i,n) bit[i]=bit[i>>1]+(i&1);
    REP(i,n) scanf("%d",&A[bit[i]][i]);
    REP(i,n) scanf("%d",&B[bit[i]][i]);
    pw1526[0]=1;
    rep(i,1,n) pw1526[i]=1526ll*pw1526[i-1]%M;
    REP(i,n) mul_(A[bit[i]][i],powMM(2,bit[i]));
    REP(i,m+1) fwt(A[i],n,1),fwt(B[i],n,1);
    REP(k,m+1) REP(i,m-k+1) {
        int t=0,j=i+k;
        REP(t,n) add_(C[k][t],(ll)A[i][t]*B[j][t]%M);
    } REP(i,m+1) fwt(C[i],n,-1);
    REP(i,n) ans[i]=C[bit[i]][i];
    int Ans=0;
    REP(i,n) add_(Ans,(ll)ans[i]*pw1526[i]%M);
    printf("%d\n",Ans);
}
```

高斯消元

indstro¨m–Gessel–Viennot lemma:

使用这个定理必须是平面图;

这个是个求不相交路径对数的方案数的定理

答案是：下列矩阵行列式，其中 A[i,j]表示 i 到 j 方案数

|A[1,1],A[1,2]|

|A[2,1],A[2,2]|

```
//求行列式的值
//%m,m 为质数的积
//从 0 开始
template<typename T>inline T poww(T a,T b,T M) {
    T ret=1;
    for (; b; b>>=1ll,a=1ll*a*a%M)
        if (b&1) ret=1ll*ret*a%M;
    return ret;
}
LL guass(LL A[107][107],int n,LL M) {
    LL ret=1; int i,j,k;
    REP(i,n) {
        int id=i;
        if (!A[i][i]) rep(j,i+1,n) if (A[j][i]) id=j;
        if (!A[id][i]) continue;
        if (id!=i) {rep(j,i,n) swap(A[i][j],A[id][j]); ret*=-1;}
        A[i][i]%=M; (A[i][i]<0) &&(A[i][i]+=M);
        LL rev=poww(A[i][i],M-2,M);
        rep(k,i+1,n)
        rrep(j,i,n)(A[k][j]-=(LL)A[k][i]*rev%M*A[i][j])%=M;
    } REP(i,n)(ret*=A[i][i])%=M;
    (ret<0) &&(ret+=M);
    return ret;
}
```

```
LL A[107][107],B[107][107];
void exgcd(LL a,LL b,LL &d,LL &x,LL &y) {
    if (!b) {d=a; x=1; y=0;}
    else {exgcd(b,a%b,d,y,x); y-=a/b*x;}
}
vector<LL> P;
vector<LL> Ans;
LL ans;
LL chinese_remainder(vector<LL> &m,vector<LL> &r) {
    int i; LL M=m[0],R=r[0];
    rep(i,1,P.size()) {
        LL x,y,d;
        exgcd(M,m[i],d,x,y);
        if ((r[i]-R)%d) return -1;
        x=(r[i]-R)/d*x%(m[i]/d);
        R+=x*M; M=M/d*m[i];
        R%=M; (R<0) &&(R+=M);
    } return R;
}
int n,m;
int i,j,k;
int main() {
    while (~scanf("%d%d",&n,&m)) {
        P.clear(); Ans.clear();
        REP(i,n)
        REP(j,n) scanf("%lld",&A[i][j]);
        for (i=2; i*i<=m; i++) if (m%i==0) {
            P.push_back(i);
            while (m%i==0) m/=i;
        } if (m!=1) P.push_back(m);
        for (int v:P) {
            REP(i,n) REP(j,n) B[i][j]=A[i][j];
            Ans.push_back((LL)guass(B,n,v));
        }
        ans=chinese_remainder(P,Ans);
        printf("%lld\n",ans);
    }
}

//辗转相除法
        REP(i,n) {
            rep(j,i+1,n) {
                int x=i,y=j;
                while (a[y][i]) {
                    LL t=a[x][i]/a[y][i];
                    rep(k,i,n) a[x][k]=(a[x][k]-a[y][k]*t)%m;
                    swap(x,y);
                }
                if (x!=i) {
                    rep(k,i,n) swap(a[i][k],a[x][k]);
                    ans=(-ans+m)%m;
                }
            }
            ans=ans*a[i][i]%m;
            ans=(ans+m)%m;
        }
```

矩阵树定理|拉格朗日插值

```
// 题意:求生成树中含 k 条给定树边的生成树个数
// 做法:为给定边加不同权值,然后矩阵树定理
// 矩阵树定理:生成树数量=|基尔霍夫矩阵 C=D-A|;
// D 为度数矩阵,A 为边矩阵
// 然后拉格朗日插值求出系数即可
LL guass(LL A[107][107],int n,LL M) {
    LL ret=1; int i,j,k;
    REP(i,n) {
        int id=i;
        if (!A[i][i]) rep(j,i+1,n) if (A[j][i]) id=j;
        if (!A[id][i]) continue;
        if (id!=i) {rep(j,i,n) swap(A[i][j],A[id][j]); ret*=-1;}
        A[i][i]%=M; (A[i][i]<0) &&(A[i][i]+=M);
        LL rev=poww(A[i][i],M-2,M);
        rep(k,i+1,n) rrep(j,i,n)
            (A[k][j]-=(LL)A[k][i]*rev%M*A[i][j])%=M;
    } REP(i,n)(ret*=A[i][i])%=M;
    (ret<0) &&(ret+=M);
    return ret;
}
int n,m;
int i,j,k;
int a[107][107]; LL A[107][107];
LL val[107],v_v[107];
LL f[107],g[107],ans[107];
int main() {
    scanf("%d",&n);
    FOR(i,1,n-1) {
        int u,v;
        scanf("%d%d",&u,&v); u--; v--;
        a[u][v]=a[v][u]=1;
        } REP(i,n) v_v[i]=i;
        REP(k,n) {
            REP(i,n) REP(j,n) A[i][j]=0;
            REP(i,n) REP(j,n) if (i!=j) {
                if (a[i][j]) A[i][j]=M-v_v[k],A[i][i]+=v_v[k];
                else A[i][j]=M-1,A[i][i]++;
            } val[k]=guass(A,n-1,M);
        }
        g[0]=1; REP(i,n)
rFOR(j,0,i)(g[j+1]+=g[j])%=M,(g[j]*=(M-v_v[i]))%=M;
        REP(k,n) {
            LL rev=1;
            rFOR(i,0,n) f[i]=(g[i+1]+f[i+1]*v_v[k]%M+M)%M;
            REP(j,n) if (j!=k)(rev*=(v_v[k]-v_v[j]))%=M;
            (rev<0) &&(rev+=M); rev=powMM(rev,M-2);
            rev=(rev*val[k])%M;
            FOR(i,0,n)(ans[i]+=(LL)f[i]*rev%M)%=M;
        } FOR(i,0,n-1) printf("%lld ",ans[i]);
    }
```

Polya 定理| Burnside 引理

```
//HDU3923; 颜色 m, 个数 n, 翻转或者置换当成一种
//ans=1/|G|*sigma{pow(k(color),m(not move point 不动点数))}
//注意特殊形式
//Burnside 引理:等价类个数 l=sum{ci(ai)},ci 是置换下的不动点数
//这个 pow 是可以变化成其他形式的
//注意,polya 定理相当于手动算了一下 Burnside 引理中不动点的个数!
int n,m;
bool mark[maxn];
int phi[maxn];
int p[maxn],tot;
int main() {
    int i,j;
    phi[1]=1;
    FOR(i,2,1000000) {
        if (!mark[i]) p[tot++]=i,phi[i]=i-1;
        REP(j,tot) {
            if (i*p[j]>1000000) break;
            //感觉上不会爆,因为是从小往筛的
            mark[i*p[j]]=1;
            if (i%p[j]==0) {phi[i*p[j]]=phi[i]*p[j]; break;}
            else phi[i*p[j]]=phi[i]*(p[j]-1);
        }
    }
    int t,T;
    scanf("%d",&T);
    FOR(t,1,T) {
        scanf("%d%d",&m,&n);
        LL all=0,cnt=0;
        // FOR(i,1,n){
        //      (all+=powMM((LL)m,gcd(n,i)))%=M;
        //      (all<0)&&(all+=M);
        // }cnt=n;
        //置换
        FOR(i,1,n) if (n%i==0) {
            (all+=(LL)powMM(m,i)*phi[n/i])%=M;
            (all<0) &&(all+=M);
        }
        cnt=n;
        //翻转
        if (n&1) {
            (all+=(LL)n*powMM(m,(n+1)/2))%=M;
            cnt+=n;
        } else {
            (all+=(LL)n/2*powMM(m,n/2))%=M;
            (all+=(LL)n/2*powMM(m,n/2+1))%=M;
            cnt+=n;
        }
//          printf("%lld %lld\n",cnt,all);
        all=all*powMM(cnt,M-2)%M;
        printf("Case #%d: %lld\n",t,all);
    }
}
```

Miller_Rabin 素性测试+pollard_rho 因数分解

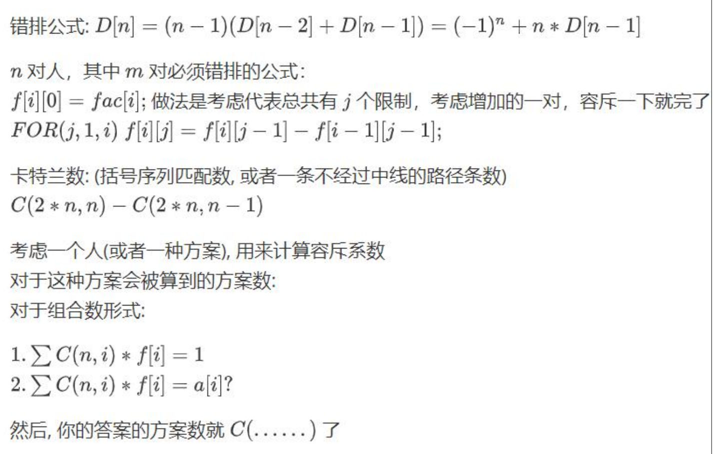
```
/*miller_rabin*/
const int times=8;// random_check; 8-12 is OK
LL mul(LL a,LL b,LL M) {
    LL ret=0;
    for (; b; b>>=1,(a+=a)>=M&&(a-=M))
        if (b&1)(ret+=a)>=M&&(ret-=M);
    return ret;
}
LL poww(LL a,LL b,LL M) {
    LL ret=1;
    for (; b; b>>=1,a=mul(a,a,M))
```

```
        if (b&1) ret=mul(ret,a,M);
    return ret;
}
bool check(LL a,LL n,LL x,LL t) {
    LL ret=poww(a,x,n);
    LL last=ret;
    for (ret=mul(ret,ret,n); t--; last=ret,ret=mul(ret,ret,n))
        if (ret==1&&last!=1&&last!=n-1) return true;
    if (ret!=1) return true;
    return false;
}
bool miller_rabin(LL n) {
    if (n<2) return false;
    if (!(n&1)) return (n==2);
    LL x=n-1,t=0;
    while (!(x&1)) x>>=1,t++;
    int i;
    REP(i,times)
    if (check(rand()%(n-1)+1,n,x,t)) return false;
    return true;
}
/*pollard_rho*/
LL pollard_rho(LL x,LL c) {
    LL x0=rand()%(x-1)+1;
    LL y=x0; c%=x;
    for (LL i=2,k=2;; i++) {
        ((x0=mul(x0,x0,x)+c)>=x)&&(x0-=x);
        LL d=gcd(y-x0+x,x);
        if (d!=1&&d!=x) return d;
        if (y==x0) return x;
        if (i==k) y=x0,k+=k;
    }
}
LL factor[107]; int tot;
void findfac(LL n,int k) {
    if (n==1) return;
    if (miller_rabin(n)) {factor[tot++]=n; return;}
    LL p=n;
    int c=k;
    while (p>=n) p=pollard_rho(p,c--);
    findfac(p,k);
    findfac(n/p,k);
}
int main() {
    int T;
    srand(time(0));
    scanf("%d",&T);
    while (T--) {
        LL n; int i;
        scanf("%I64d",&n);
        if (miller_rabin(n)) puts("Prime");
        else {
            tot=0;
            findfac(n,107);
            LL ans=factor[0];
            REP(i,tot) ans=min(ans,factor[i]);
            printf("%I64d\n",ans);
        }
    }
}
```

中国剩余定理(不一定互质)

```
void exgcd(LL a,LL b,LL &d,LL &x,LL &y){
    if (!b) {d=a;x=1;y=0;}
    else {exgcd(b,a%b,d,y,x);y-=a/b*x;}
}
int n,m;
int i,j,k;
vector<LL> P,O;
int ans;
LL chinese_remainder(vector<LL> &m,vector<LL> &r){
    int i;LL M=m[0],R=r[0];
    rep(i,1,P.size()){
        LL x,y,d;
        exgcd(M,m[i],d,x,y);
        if ((r[i]-R)%d) return -1;
        x=(r[i]-R)/d*x%(m[i]/d);
        R+=x*M;M=M/d*m[i];
        R%=M;(R<0)&&(R+=M);
    }return R;
}
int main(){
    while (~scanf("%d",&n)){
        P.clear();O.clear();
        REP(i,n){
            LL k;
            scanf("%lld",&k);P.push_back(k);
            scanf("%lld",&k);O.push_back(k);
        }printf("%lld\n",chinese_remainder(P,O));
        }
    }
```

广义容斥



错排公式: $D[n] = (n-1)(D[n-2] + D[n-1]) = (-1)^n + n * D[n-1]$

n 对人，其中 m 对必须错排的公式:
$f[i][0] = fac[i]$; 做法是考虑代表总共有 j 个限制，考虑增加的一对，容斥一下就完了
$FOR(j,1,i)\ f[i][j] = f[i][j-1] - f[i-1][j-1]$;

卡特兰数: (括号序列匹配数, 或者一条不经过中线的路径条数)
$C(2*n, n) - C(2*n, n-1)$

考虑一个人(或者一种方案), 用来计算容斥系数
对于这种方案会被算到的方案数:
对于组合数形式:

1. $\sum C(n,i) * f[i] = 1$
2. $\sum C(n,i) * f[i] = a[i]$?

然后, 你的答案的方案数就 $C(\ldots\ldots)$ 了

Prime-counting function

//这道题题意:小于 n 有多少个数字有 4 个因子

//(两个质数积,一个质数三次方)

//注意容斥减去多算的

//http://codeforces.com/blog/entry/44466?#comment-290036/

//考虑 S(v,m):2...v,质因子全都>=m;那么考虑容斥:

//容斥掉的至少有一个 p,而且没有小于 p 的因子

//很明显的,p=min(p,sqrt(v));

//S(v,p)=S(v,p-1)-(S(v/p,p-1)-S(p-1,p-1));(DP)

//那么反过来算即可;pi(n)=S(n,n);

//H[i]:pi(n/i);L[i]:pi(i)

//计算过程中,L[i]表示 S(i,p),最终 S(i,i)

//简单的这样 DP,时间复杂度 O(n^3/4),如果预处理 n^2/3 则最终 n^2/3

//在后方,如果要容斥,FOR 是很不方便的,感觉还是最好直接搞复杂度有保障

Min_25 筛

SPOJ DIVCNTK(sum_ \sigma(i^k))

```
namespace seives { // 抄的 define
#define clr(ar) memset(ar, 0, sizeof(ar))
#define chkbit(ar, i) (((ar[(i) >> 6]) & (1 << (((i) >> 1) & 31))))
#define setbit(ar, i) (((ar[(i) >> 6]) |= (1 << (((i) >> 1) & 31))))
#define isprime(x) (( (x) && ((x)&1) && (!chkbit(ar, (x)))) || ((x) == 2))
    const int MAXP=66666;
    const int MAX=100010;//euler_seive
    const int maxn=100010;//min_25, =sqrt(n)
    int p[MAXP],tot;
    ui ar[(MAX>>6)+7]= {0};
    void init() {//seives
        setbit(ar,0); setbit(ar,1);
        int i,j; tot=0;
        rep(i,2,MAX) {
            if (isprime(i)) p[tot++]=i;
            REP(j,tot) {
                if (i*p[j]>=MAX) break;
                if ((i*p[j])&1) setbit(ar,i*p[j]);
                if (i%p[j]==0) break;
            }
        }
    }
    // 普通 pcf 公式: g(i,j)=g(i-1,j)-p^k*g(i-1,j/p)
    // 只有小于等于 sqrt 的 p 有用，所以枚举这个，考虑对其他答案的贡献
    // 对于某个积性函数: (算贡献)
    // g(i,j)=g(i-1,j)+\sum_p^k F(p^k)*g(i-1,j/[p^k]),还要加p^k 的贡献
    // 对于小于等于 sqrt 的 p，直接筛
    // 对于大于的，贡献只会是 F(p)! 也就是...直接洲阁筛把答案的贡献加进去
    // 这个加贡献=_= 竟然是直接 pcf 求个前缀和啥的就完事了啊=_=
    // typedef ull ll;
    // 注意如果想要去掉某个质数的贡献，这个 p[k]至少要筛到sqrtn...
    // 注意 F1 的贡献，是要乘的...
    // 我这个 F 和 G 和一般的定义是反的...要先算 G
    // F 和 G 定义是质数处的前缀和
    // getans 处记得如果质数贡献不同得改
    ll n,m;//blocksize
    ll H[maxn],L[maxn];
    void pcf() {
        ll p,k;
        FOR(p,1,m) L[p]=p-1,H[p]=n/p-1;
        FOR(p,2,m) {
            if (L[p]==L[p-1]) continue;//not_prime
            FOR(k,1,min(m,n/p/p)) {
                if (p*k<=m) H[k]-=H[p*k]-L[p-1];
                else H[k]-=L[n/p/k]-L[p-1];
            } rFOR(k,p*p,m) L[k]-=L[k/p]-L[p-1];
        }
    }
    ll F[maxn],G[maxn];//F[n/k]:H[n/k], G[i]:L[i]
    ll K;
    ll getans(ll x,int i) {
        if (x<=1||p[i]>x) return 0;
        if (p[i]>m) return F[n/x]-G[m];
        ll ans=((x<=m)?G[x]:F[n/x])-G[p[i]-1];
        for (; (ll)p[i]*p[i]<=x; i++) {
            for (ll _x=x/p[i],c=1; _x>=p[i]; _x/=p[i],c++)
                ans+=getans(_x,i+1)*(c*K+1)+((c+1)*K+1);
        } return ans;
    }
    ll solve() {
        int p;
        for (m=1; m*m<=n; ++m); m--; pcf();
        FOR(p,1,m) F[p]=H[p]*(K+1),G[p]=L[p]*(K+1);
        return getans(n,0)+1;//1:1
    }
}
```

取模的质数个数筛的方式:

```
    ll n,m;//blocksize
    ll H[maxn][4],L[maxn][4];
    // p%4==1 那么 p 可以表示为两个平方因子的和
    void pcf() {
        ll p,k; int i;
        FOR(p,1,m) {
            REP(i,4) L[p][i]=(p-i+4)/4;
            REP(i,4) H[p][i]=(n/p-i+4)/4;
            REP(i,2) H[p][i%4]--,L[p][i%4]--;
        }
        FOR(p,2,m) {
            if (L[p][p%4]==L[p-1][p%4])
continue;//not_prime
            static int nxt[4];
            REP(i,4) nxt[i]=i*p%4;
            FOR(k,1,min(m,n/p/p)) {
                if (p*k<=m) REP(i,4)
H[k][nxt[i]]-=H[p*k][i]-L[p-1][i];
                else REP(i,4)
H[k][nxt[i]]-=L[n/p/k][i]-L[p-1][i];
            } rFOR(k,p*p,m) REP(i,4)
L[k][nxt[i]]-=L[k/p][i]-L[p-1][i];
        }
    }
```

积性函数 前缀和 杜教筛

$n=\sum_{d|n}\varphi(d)$ 将phi看作容斥系数
$[n=1]=\sum_{d|n}\mu(d)$ 将 $\frac{i}{n}$化为最简分数
$\varphi(n)=\sum_{d|n}\frac{n}{d}*\mu(d)$

$\frac{\varphi(n)}{n}=\sum_{d|n}\frac{\mu(d)}{d}$ (这个拆开证明)

这里可以把gcd或者lcm的式子提出来!
(重要!) 1...n的与n互质数和=($\frac{n*\varphi(n)+[n=1]}{2}$)
然后，经过推导可能将某些式子化成简单形式就能做了qwq完全不会，智商不够没办法......

$\sum$ 的是前缀和，带下标的特殊

$\sum[gcd(i,j)=1]=\sum\mu(d)*\frac{n}{d}^2$
$\sum gcd(i,n)=\sum_{d|n}d*phi(\frac{n}{d})$
$\sum$ 约数个数$\sigma(n)=\sum\frac{n}{d}$

$\sigma(n*m)=\sum_{i|n}\sum_{j|m}[gcd(i,j)=1]$(原因是枚举约数$i*\frac{m}{j}$, $gcd(i,j)=1$不会算重)
$\sum_i\sum_j\sigma(i*j)=\sum_d\mu(d)*\sum_i\frac{n}{d*i}*\sum_j\frac{n}{d*j}$
$\sum$ 约数和$\sigma_1(i)=\sum\frac{n}{d}*d=\sum\frac{(\frac{n}{d})*((\frac{n}{d})+1)}{2}$
$\mu(n)^2=0$(无平方因子)时, 存在$\varphi(n*k)=\sum_{d|gcd(n,k)}\varphi(d)\varphi(k)$
$\sum_i^n\mu(i)^2=\sum_i\mu(i)*\frac{n}{i*i}$(可以认为是无平方因子数个数)
注意最好还是化成能书写的形式，脑补还是很可能出问题!

关于莫比乌斯反演:
$f(n)=\sum_{d|n}g(d)$ 等价于 $g(n)=\sum_{d|n}\mu(d)f(\frac{n}{d})$
本质是个容斥

关于积性函数:
单位函数 $e(x)=[x==1]$
常函数$I(x)=1$
幂函数$id^k(x)=x^k$
欧拉函数$phi(x)=x*\prod(1-\frac{1}{p})$
莫比乌斯函数 $\mu(x)=(-1)^k, x=p1*p2*...*pk$
约数个数函数 $\sigma(d)=\prod_{p|d}(k+1)$
约数和函数 $\sigma_1(d)=\prod_{p|d}(p*k+1)$

狄利克雷卷积: $(f*g)(n)=\sum_{d|n}f(d)*g(\frac{n}{d})$
积性函数的狄利克雷卷积也是积性函数
可以将一个ans化成多个狄利克雷卷积相加的形式
(重要!) 狄利克雷卷积满足交换律、结合律，对加法满足分配律

积性函数前缀和（杜教筛）：
如果能通过狄利克雷卷积构造一个更好计算前缀和的函数，且用于卷积的另一个函数也易计算，则可以简化计算过程。
你需要先构造一个可以很快计算前缀和的东西, 然后利用交换 i 和 d | i 来化简式子来加速运算

这个 x 可能非常大, 乘起来就可能会爆掉, 需要特别注意!
可以不用map来记录比较大的数的答案, 可以开个数组直接记录g(i)代表n/i的答案

其他奇怪的东西:
rng_58-clj等式
$\sum_i^a\sum_j^b\sum_k^c d(i*j*k)=\sum[gcd(i,j)=gcd(j,k)=gcd(k,i)=1]$

$*\frac{i}{a}\frac{j}{b}\frac{k}{c}$

这个可以扩展到任意维度

Zoj 3881

$\sum_i^n\sum_{d|i}rad(d)*\varphi(\frac{d}{rad(d)})$ rad表示最大无平方因子数

tls: 后面的这个东西很明显是个积性函数! 所以就不用努力了=_=

假设 $p^k|i$

$=\sum_i^n\prod_{p|i,p是质数}(1+\sum_t^k p*\varphi(p^{t-1}))$ 后面这个东西按t=1分类

$=\sum_i^n\prod_{p|i,p是质数}(1+p^k)$

tls: 所以后面这个东西要么全选要么全不选

$=\sum_i^n\sum_{k|i}[gcd(k,\frac{i}{k})=1]*k$

let j=i/k

$=\sum_j^{\frac{n}{k}}\sum_k^n[gcd(k,j)=1]*k$

$=\sum_j^{\frac{n}{k}}\sum_k^n\sum_{d|gcd(k,j)}\mu(d)*k$

$=\sum_d\mu(d)*d*\sum_j^{\frac{n}{k*d^2}}\sum_k^{\frac{n}{d}}*k$

$=\sum_d\mu(d)*d*\sum_k^{\frac{n}{d^2}}*k*\frac{n}{k*d^2}$

感谢 tls 教我不要这么写了。。这个界限还是写个乘积的形式为妙

后半段是 $\sum_i^{\frac{n}{d^2}}\sigma_1$ 而且直接就可以求，就做完了

```
vector<int> P[maxn];
namespace seives { // 抄的 define
#define clr(ar) memset(ar, 0, sizeof(ar))
    #define chkbit(ar, i) (((ar[(i) >> 6]) & (1 << (((i) >> 1) & 31))))
    #define setbit(ar, i) (((ar[(i) >> 6]) |= (1 << (((i) >> 1) & 31))))
    #define isprime(x) (( (x) && ((x)&1) && (!chkbit(ar, (x)))) ||
((x) == 2))
        const int MAXP=666666;
        const int MAX=2000010;
        int mu[MAX],sigma1[MAX],c1[MAX],f[MAX];
        int p[MAXP],tot;
        ui ar[(MAX>>6)+7]= {0};
        void init() {
            setbit(ar,0); setbit(ar,1);
            int i,j; tot=0; mu[1]=1; sigma1[1]=1;
            rep(i,2,MAX) {
                if (isprime(i))
p[tot++]=i,mu[i]=-1,sigma1[i]=i+1,c1[i]=i+1;
                REP(j,tot) {
                    if (i*p[j]>=MAX) break;
                    if ((i*p[j])&1) setbit(ar,i*p[j]);
                    if (i%p[j]==0) {
                        c1[i*p[j]]=p[j]*c1[i]+1;

sigma1[i*p[j]]=sigma1[i]/c1[i]*c1[i*p[j]];
                        break;
                    } else {
                        c1[i*p[j]]=p[j]+1;
                        sigma1[i*p[j]]=sigma1[i]*(p[j]+1);
                        mu[i*p[j]]=-mu[i];
                    }
                }
            } rep(i,1,MAX) f[i]=sigma1[i],add_(f[i],f[i-1]);
        }
        map<int,int> HASH;
        int get2(ll x){
            x%=M; return x*(x+1)%M*500000005%M;
        }
        int get_f(ll x){//直接 sqrt 也行
            if (x<MAX) return f[x];
            if (HASH.count(x)) return HASH[x];
            ll ret=0; ll l;
            FOR(l,1,x) {
                ll t=x/l,r=x/t;
                add_(ret,(get2(r)-get2(l-1)+M)%M*(t%M)%M);
                l=r;
            } return HASH[x]=ret;
        }
}
int main() {
    // startTimer();
    seives::init(); ll n;
    // printTimer();
    while (~scanf("%lld",&n)){
        // startTimer();
        int ans=0;
        for (ll d=1;d*d<=n;d++)
add_(ans,(M+seives::mu[d])*d%M*seives::get_f(n/d/d)%M);
        printf("%d\n",ans);
        // printTimer();
    }
}
```

类欧几里得

一定注意前面是 a,后面是 b,线段树一定要注意顺序

f(a,b,c,n)=sigma{(ai+b)/c}; (0->n)

g(a,b,c,n)=sigma{(ai+b)/c*i}; (0->n)

h(a,b,c,n)=sigma{((ai+b)/c)^2}; (0->n)

let m=(a*n+b)/c;

推导 f:

a=0:

return b/c*(n+1)

a>=c||b>=c:有一部分是规律的;

return (a/c)*n(n+1)/2+(b/c)*(n+1)+f(a%c,b%c,c,n)

else:直接算,这个东西是个梯形中的点数,反过来算就可以了

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [(ai+b)/c>=j+1]

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [ai>=cj+c−b]

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [ai>cj+c−b−1]

f(a,b,c,n)=∑i=0->n ∑j=0->m-1 [i>(cj+c−b−1)/a]

f(a,b,c,n)=∑j=0->m (n−(cj+c−b−1)/a)

f(a,b,c,n)=n*m-f(c,c-b-1,a,m-1);

推导 g:

a=0:

return b/c*n(n+1)/2 (sigma 的是 i)

a>=c||b>=c:有一部分是规律的;

g(a,b,c,n)=(a/c)*n(n+1)(2n+1)/6+(b/c)*n(n+1)/2+g(a%c,b%c,c,n)

else:

g(a,b,c,n)=∑i=0->n i*∑j=0->m [(ai+b)/c>=j]

g(a,b,c,n)=∑i=0->n i*∑j=0->m-1 [i>(cj+c−b−1)/a]

然后把这个 i 放进去求和

g(a,b,c,n)=1/2*∑j=0->m-1 (n+1+(cj+c−b−1)/a)*(n−(cj+c−b−1)/a)

g(a,b,c,n)=1/2*∑j=0->m-1 n(n+1)−(cj+c−b−1)/a−[(cj+c−b−1)/a]^2

g(a,b,c,n)=1/2*[n(n+1)*m − f(c,c − b − 1,a,m − 1) − h(c,c − b − 1,a,m−1)]

推导 h:

a=0:

return (b/c)^2*(n+1) (sigma 的是 i)

a>=c||b>=c:有一部分是规律的;

h(a,b,c,n)=(a/c)^2*n(n+1)(2n+1)/6+(b/c)^2*(n+1)+(a/c)*(b/c)*n(n+1)

+h(a%c,b%c,c,n)+2*(a/c)*g(a%c,b%c,c,n)+2*(b/c)*f(a%c,b%c,c,n)

else:

n^2=2*n(n+1)/2−n=2(∑i=0->n i)−n

有了思路我们来推 h

h(a,b,c,n)=∑i=0->n (2(∑j=1->(ai+b)/c j)−(ai+b)/c)

可以想到交换主体。

h(a,b,c,n)=∑j=0->m-1 (j+1)*∑i=0->n [(ai+b)/c>=j+1] − f(a,b,c,n)

h(a,b,c,n)=∑j=0->m-1 (j+1)*∑i=0->n [i>(cj+c − b − 1)/a] −f(a,b,c,n)

h(a,b,c,n)=∑j=0->m-1 (j+1)*(n−(cj+c−b−1)/a)−f(a,b,c,n)

h(a,b,c,n)=n*m(m+1)−2g(c,c−b−1,a,m−1)−2f(c,c−b−1,a,m−1)−f(a,b,c,n)

```
    // 题意:n%1,n%2...异或, 做法是 BSGS 然后类欧几里得
    // 每块是 n-n/l*l ^ ... ^ n-n/r*r
    // 也就是 n-(n/l)*k,(等价于 n%r+(n/l)*k) k 是 0->r-l
    // 按位计算, 就变成了个类欧几里得
    // 玄学卡常,n<=某值直接暴力, 这里 tls 说是一个 log 的
    LL f(LL a,LL b,LL c,LL n) {
        if (a==0) return b/c*(n+1);
        if (a>=c||b>=c) return
(a/c)*n*(n+1)/2+(b/c)*(n+1)+f(a%c,b%c,c,n);
        LL m=(a*n+b)/c;
        return n*m-f(c,c-b-1,a,m-1);
    }
    LL solve(LL l,LL c,LL n) {
        LL ret=0,i;
        if (n<=10000) REP(i,n+1) ret^=l,l+=c;
        else REP(i,40) ret^=(f(c,l,(1ll<<i),n)&1)<<i;
    return ret;
}
LL getans(LL n) {
    LL ans=0;
    for (LL l=1,r; l<=n;) {
        r=n/(n/l);
        ans^=solve(n%r,n/l,r-l);
        l=r+1;
    } return ans;
}
int main() {
    int T;
    int i,j,k;
    scanf("%d",&T);
    while (T--) {
        LL n;
        scanf("%lld",&n);
        printf("%lld\n",getans(n));
    }
    return 0;
}
```

欧拉降幂公式

//n^x(mod m)=m^(phi(m)+x%phi(m))%m (x>m)
//这个题让求 pow(l,pow(l+1...pow(r)))

```
inline int mod(LL a,int b){
    if (a<b) return a;
    return a%b+b;
}
inline int poww(int a,int b,int M){
    int ret=1;
    for (;b;b>>=1ll,a=mod(1ll*a*a,M))
        if (b&1) ret=mod(1ll*ret*a,M);
    return ret;
}
typedef pair<int,int> pii;
int P[maxn];
int phi(int x){
    int k=x;
    for (int i=2;i*i<=k;i++) if (k%i==0){
        x=x/i*(i-1);
        while (k%i==0) k/=i;
    }if (k!=1) x=x/k*(k-1);
    return x;
}
int a[maxn];
int tot;
int solve(int l,int r,int pos){
    if (l==r||pos==tot) return mod(a[l],P[pos]);
    return poww(a[l],solve(l+1,r,pos+1),P[pos]);
}
int n,m,q,i,j,k;
int main(){
    scanf("%d%d",&n,&m);
    FOR(i,1,n) scanf("%d",&a[i]);
    P[1]=m;
    for (tot=1;P[tot]!=1;tot++) P[tot+1]=phi(P[tot]);
    scanf("%d",&q);
    FOR(i,1,q){
        int l,r;int ans=1;
        scanf("%d%d",&l,&r);
        printf("%d\n",solve(l,r,1)%m);
    }
}
```

单纯形法

注意，只能做实数规划

输出方案:

```
//http://www.voidcn.com/article/p-kkqovyic-qh.html
//m 个式子,n 个变量
//maximize A[0][i]*value
//sigma<=A[i][0]
namespace Simplex {
    int n,m;//n 变量, m 式子
    const int maxn=500,maxm=5000;
    double A[maxm][maxn];//
    int id[maxn+maxm];//base
    const double inf=1e20;
    const double eps=1e-7;
    void pivot(int l,int e) {
        int tt=id[n+l]; id[n+l]=id[e]; id[e]=tt;
        int i,j; double t=A[l][e]; A[l][e]=1;
        FOR(j,0,n) A[l][j]/=t;
        FOR(i,0,m) if (i!=l && abs(A[i][e])>eps) {
            t=A[i][e]; A[i][e]=0;
            for (j=0; j<=n; j++)
                A[i][j]-=A[l][j]*t;
            }
        }
        bool initialize() {
            int i,j;
            FOR(i,1,n) id[i]=i;
            while (1) {
                int e=0,l=0;
                FOR(i,1,m) if (A[i][0]<-eps && (!l || (rand()&1)))
l=i;
                if (!l) break;
                FOR(j,1,n) if (A[l][j]<-eps && (!e || (rand()&1)))
e=j;
                if (!e) return 0;//Infeasible,无解
                pivot(l,e);
            } return 1;
        }
        double ans[maxn],value;
        bool simplex() {
            int i,j;
            while (true) {
                int l=0,e=0; double minn=inf;
                FOR(j,1,n) if (A[0][j]>eps) {e=j; break;}
                if (!e) break;
                FOR(i,1,m) if (A[i][e]>eps &&
A[i][0]/A[i][e]<minn)
                    minn=A[i][0]/A[i][e],l=i;
                if (!l) return 0;//Unbounded,inf
                pivot(l,e);
            }
            FOR(i,1,m) ans[id[n+i]]=A[i][0];
            value=-A[0][0];//maxvalue
            return 1;
        }
    }
    int main() {
        int n,m,i,j;
        scanf("%d%d",&n,&m);
        FOR(i,1,n) scanf("%lf",&Simplex::A[0][i]);
        FOR(i,1,m) {
            int l,r,c;
            scanf("%d%d%d",&l,&r,&c);
            FOR(j,l,r) Simplex::A[i][j]=1;
            Simplex::A[i][0]=c;
        }
        Simplex::n=n; Simplex::m=m;
        assert(Simplex::initialize());
        assert(Simplex::simplex());
        // FOR(i,1,n) printf("%.0f ",Simplex::ans[i]); puts("");
        printf("%.0f",Simplex::value);
    }
```

不输出方案，另一种写法：

```
//m 个式子,n 个变量
//maximize C[i]*value
namespace Simplex {
    int n,m;//n 变量，m 式子
    const int maxn=500,maxm=5000;
    double A[maxm][maxn],B[maxm];
    double C[maxn];//base
    double v=0;
    const double inf=1e20;
    const double eps=1e-7;
    void pivot(int l,int e) {
        B[l]/=A[l][e];
        for (int i=1; i<=n; i++)
            if (i!=e) A[l][i]/=A[l][e];
        A[l][e]=1/A[l][e];
        for (int i=1; i<=m; i++)
            if (i!=l&&fabs(A[i][e])>eps) {
                B[i]-=B[l]*A[i][e];
                for (int j=1; j<=n; j++)
                    if (j!=e) A[i][j]-=A[l][j]*A[i][e];
                A[i][e]=-A[l][e]*A[i][e];
            }
        v+=C[e]*B[l];
        for (int i=1; i<=n; i++)
            if (i!=e) C[i]-=C[e]*A[l][i];
        C[e]=-C[e]*A[l][e];
    }
    double simplex() {
        int l,e;
        double t;
        while (true) {
            e=n+1;
            for (int i=1; i<=n; i++)
                if (C[i]>eps) {e=i; break;}
            if (e==n+1) break;
            t=inf; l=0;
            for (int i=1; i<=m; i++)
                if (A[i][e]>eps&&t>B[i]/A[i][e]) {
                    t=B[i]/A[i][e]; l=i;
                }
                if (t==inf) return inf;
                pivot(l,e);
            }
            return v;
        }
}
int main() {
    int n,m,i,j;
    scanf("%d%d",&n,&m);
    FOR(i,1,n) scanf("%lf",&Simplex::C[i]);
    FOR(i,1,m) {
        int l,r,c;
        scanf("%d%d%d",&l,&r,&c);
        FOR(j,l,r) Simplex::A[i][j]=1;
        Simplex::B[i]=c;
    }
    Simplex::n=n; Simplex::m=m;
    printf("%.0f",Simplex::simplex());
}
```

其他的东西

杜教线性递推 BM 板子

```
    int _n;
    namespace linear_seq {
        const int N=10010;
        ll res[N],base[N],_c[N],_md[N];
        vector<int> Md;
        void mul(ll *a,ll *b,int k) {
            rep(i,0,k+k) _c[i]=0;
            rep(i,0,k) if (a[i]) rep(j,0,k)
_c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
            for (int i=k+k-1;i>=k;i--) if (_c[i])
                rep(j,0,SZ(Md))
_c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
            rep(i,0,k) a[i]=_c[i];
        }
        int solve(ll n,VI a,VI b) { // a 系数 b 初值
b[n+1]=a[0]*b[n]+...
    //          printf("%d\n",SZ(b));
            ll ans=0,pnt=0;
            int k=SZ(a);
            assert(SZ(a)==SZ(b));
            rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
            Md.clear();
            rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
            rep(i,0,k) res[i]=base[i]=0;
            res[0]=1;
            while ((1ll<<pnt)<=n) pnt++;
            for (int p=pnt;p>=0;p--) {
                mul(res,res,k);
                if ((n>>p)&1) {
                    for (int i=k-1;i>=0;i--)
res[i+1]=res[i];res[0]=0;
                    rep(j,0,SZ(Md))
res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
                }
            }
            rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
            if (ans<0) ans+=mod;
            return ans;
        }
        VI BM(VI s) {
            VI C(1,1),B(1,1);
            int L=0,m=1,b=1;
            rep(n,0,SZ(s)) {
                ll d=0;
                rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
                if (d==0) ++m;
                else if (2*L<=n) {
                    VI T=C;
                    ll c=mod-d*powmod(b,mod-2)%mod;
                    while (SZ(C)<SZ(B)+m) C.pb(0);
                    rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                    L=n+1-L; B=T; b=d; m=1;
                } else {
                    ll c=mod-d*powmod(b,mod-2)%mod;
                    while (SZ(C)<SZ(B)+m) C.pb(0);
                    rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                    ++m;
                }
            }
            return C;
        }
        int gao(VI a,ll n) {
            VI c=BM(a);
            c.erase(c.begin());
            rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
            for (int v:c) printf("%d ",v);puts("");
            return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
        }
    };

    int main() {
        int
k=linear_seq::gao(VI{7,16,25,50,84,159,277,511,906,1651,2
952,5348,9601,17345,31199,56288,101341},10);
        printf("%d\n",k);
        for (scanf("%d",&_);_;_--) {
            scanf("%d",&n);

printf("%d\n",linear_seq::gao(VI{0,1,1,2,3,5,8,13,21,34},n-1));
        }
    }
```

任意模数 BM 板子

```
    #include <bits/stdc++.h>
    using namespace std;
    #ifndef ONLINE_JUDGE
    #define debug(fmt, ...) fprintf(stderr, "[%s] " fmt "\n", __func__,
##__VA_ARGS__)
    #else
    #define debug(...)
    #endif

    // given first m items init[0..m-1] and coefficents
trans[0..m-1] or
    // given first 2 *m items init[0..2m-1], it will compute
trans[0..m-1]
    // for you. trans[0..m] should be given as that
    //        init[m] = sum_{i=0}^{m-1} init[i] * trans[i]
    struct LinearRecurrence {
        using int64 = long long;
        using vec = std::vector<int64>;

        static void extand(vec &a, size_t d, int64 value = 0) {
            if (d <= a.size()) return;
            a.resize(d, value);
        }
        static vec BerlekampMassey(const vec &s, int64 mod) {
            std::function<int64(int64)> inverse = [&](int64 a) {
                return a == 1 ? 1 : (int64)(mod - mod / a) *
inverse(mod % a) % mod;
            };
            vec A = {1}, B = {1};
            int64 b = s[0];
            for (size_t i = 1, m = 1; i < s.size(); ++i, m++) {
                int64 d = 0;
                for (size_t j = 0; j < A.size(); ++j) {
                    d += A[j] * s[i - j] % mod;
                }
                if (!(d %= mod)) continue;
                if (2 * (A.size() - 1) <= i) {
                    auto temp = A;
                    extand(A, B.size() + m);
                    int64 coef = d * inverse(b) % mod;
                    for (size_t j = 0; j < B.size(); ++j) {
                        A[j + m] -= coef * B[j] % mod;
                        if (A[j + m] < 0) A[j + m] += mod;
                    }
                    B = temp, b = d, m = 0;
                } else {
                    extand(A, B.size() + m);
                    int64 coef = d * inverse(b) % mod;
                    for (size_t j = 0; j < B.size(); ++j) {
                        A[j + m] -= coef * B[j] % mod;
                        if (A[j + m] < 0) A[j + m] += mod;
                    }
                }
            }
            return A;
        }
        static void exgcd(int64 a, int64 b, int64 &g, int64 &x,
int64 &y) {
            if (!b)
                x = 1, y = 0, g = a;
            else {
                exgcd(b, a % b, g, y, x);
                y -= x * (a / b);
            }
        }
        static int64 crt(const vec &c, const vec &m) {
            int n = c.size();
            int64 M = 1, ans = 0;
            for (int i = 0; i < n; ++i) M *= m[i];
            for (int i = 0; i < n; ++i) {
                int64 x, y, g, tm = M / m[i];
                exgcd(tm, m[i], g, x, y);
                ans = (ans + tm * x * c[i] % M) % M;
            }
            return (ans + M) % M;
        }
        static vec ReedsSloane(const vec &s, int64 mod) {
            auto inverse = [](int64 a, int64 m) {
                int64 d, x, y;
            exgcd(a, m, d, x, y);
            return d == 1 ? (x % m + m) % m : -1;
        };
        auto L = [](const vec& a, const vec& b) {
            int da = (a.size() > 1 || (a.size() == 1 && a[0])) ?
a.size() - 1 : -1000;
            int db = (b.size() > 1 || (b.size() == 1 && b[0])) ?
b.size() - 1 : -1000;
            return std::max(da, db + 1);
        };
        auto prime_power = [&](const vec& s, int64 mod,
int64 p, int64 e) {
            // linear feedback shift register mod p^e, p is
prime
            std::vector<vec> a(e), b(e), an(e), bn(e), ao(e),
bo(e);
            vec t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
            ;
            pw[0] = 1;
            for (int i = pw[0] = 1; i <= e; ++i) pw[i] = pw[i - 1]
* p;
            for (int64 i = 0; i < e; ++i) {
                a[i] = {pw[i]}, an[i] = {pw[i]};
                b[i] = {0}, bn[i] = {s[0] * pw[i] % mod};
                t[i] = s[0] * pw[i] % mod;
                if (t[i] == 0) {
                    t[i] = 1, u[i] = e;
                } else {
                    for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i])
                        ;
                }
            }
            for (size_t k = 1; k < s.size(); ++k) {
                for (int g = 0; g < e; ++g) {
                    if (L(an[g], bn[g]) > L(a[g], b[g])) {
                        ao[g] = a[e - 1 - u[g]];
                        bo[g] = b[e - 1 - u[g]];
                        to[g] = t[e - 1 - u[g]];
                        uo[g] = u[e - 1 - u[g]];
                        r[g] = k - 1;
                    }
                }
                a = an, b = bn;
                for (int o = 0; o < e; ++o) {
                    int64 d = 0;
                    for (size_t i = 0; i < a[o].size() && i <= k; ++i)
{
                        d = (d + a[o][i] * s[k - i]) % mod;
                    }
                    if (d == 0) {
                        t[o] = 1, u[o] = e;
                    } else {
                        for (u[o] = 0, t[o] = d; t[o] % p == 0;
t[o] /= p, ++u[o])
                            ;
                        int g = e - 1 - u[o];
                        if (L(a[g], b[g]) == 0) {
                            extand(bn[o], k + 1);
                            bn[o][k] = (bn[o][k] + d) % mod;
                        } else {
                            int64 coef = t[o] * inverse(to[g],
mod) % mod * pw[u[o] - uo[g]] % mod;
                            int m = k - r[g];
                            extand(an[o], ao[g].size() + m);
                            extand(bn[o], bo[g].size() + m);
                            for (size_t i = 0; i < ao[g].size(); ++i)
{
                                an[o][i + m] -= coef *
ao[g][i] % mod;
                                if (an[o][i + m] < 0) an[o][i +
m] += mod;
                            }
                            while (an[o].size() && an[o].back()
== 0) an[o].pop_back();
                            for (size_t i = 0; i < bo[g].size(); ++i)
{
                                bn[o][i + m] -= coef *
bo[g][i] % mod;
                                if (bn[o][i + m] < 0) bn[o][i +
m] -= mod;
                            }
                            while (bn[o].size() && bn[o].back()
== 0) bn[o].pop_back();
                        }
                    }
                }
            }
            return std::make_pair(an[0], bn[0]);
        };
```

```
            std::vector<std::tuple<int64, int64, int>> fac;
            for (int64 i = 2; i * i <= mod; ++i) {
                if (mod % i == 0) {
                    int64 cnt = 0, pw = 1;
                    while (mod % i == 0) mod /= i, ++cnt, pw *= i;
                    fac.emplace_back(pw, i, cnt);
                }
            }
            if (mod > 1) fac.emplace_back(mod, mod, 1);
            std::vector<vec> as;
            size_t n = 0;
            for (auto&& x : fac) {
                int64 mod, p, e;
                vec a, b;
                std::tie(mod, p, e) = x;
                auto ss = s;
                for (auto&& x : ss) x %= mod;
                std::tie(a, b) = prime_power(ss, mod, p, e);
                as.emplace_back(a);
                n = std::max(n, a.size());
            }
            vec a(n), c(as.size()), m(as.size());
            for (size_t i = 0; i < n; ++i) {
                for (size_t j = 0; j < as.size(); ++j) {
                    m[j] = std::get<0>(fac[j]);
                    c[j] = i < as[j].size() ? as[j][i] : 0;
                }
                a[i] = crt(c, m);
            }
            return a;
        }

        LinearRecurrence(const vec &s, const vec &c, int64 mod) : init(s), trans(c), mod(mod), m(s.size()) {}
        LinearRecurrence(const vec &s, int64 mod, bool is_prime = true) : mod(mod) {
            vec A;
            if (is_prime)
                A = BerlekampMassey(s, mod);
            else
                A = ReedsSloane(s, mod);
            if (A.empty()) A = {0};
            m = A.size() - 1;
            trans.resize(m);
            for (int i = 0; i < m; ++i) {
                trans[i] = (mod - A[i + 1]) % mod;
            }
            std::reverse(trans.begin(), trans.end());
            init = {s.begin(), s.begin() + m};
        }
        int64 calc(int64 n) {
            if (mod == 1) return 0;
            if (n < m) return init[n];
            vec v(m), u(m << 1);
            int msk = !!n;
            for (int64 m = n; m > 1; m >>= 1) msk <<= 1;
            v[0] = 1 % mod;
            for (int x = 0; msk; msk >>= 1, x <<= 1) {
                std::fill_n(u.begin(), m * 2, 0);
                x |= !!(n & msk);
                if (x < m)
                    u[x] = 1 % mod;
                else {
                    // can be optimized by fft/ntt
                    for (int i = 0; i < m; ++i) {
                        for (int j = 0, t = i + (x & 1); j < m; ++j, ++t) {
                            u[t] = (u[t] + v[i] * v[j]) % mod;
                        }
                    }
                    for (int i = m * 2 - 1; i >= m; --i) {
                        for (int j = 0, t = i - m; j < m; ++j, ++t) {
                            u[t] = (u[t] + trans[j] * u[i]) % mod;
                        }
                    }
                }
                v = {u.begin(), u.begin() + m};
            }
            int64 ret = 0;
            for (int i = 0; i < m; ++i) {
                ret = (ret + v[i] * init[i]) % mod;
            }
            return ret;
        }

        vec init, trans;
        int64 mod;
        int m;
    };
```

```
const int mod = 1e9;

typedef long long ll;

ll Pow(ll a, ll n, ll mod) {
    ll t = 1;
    for (; n; n >>= 1, (a *= a) %= mod)
        if (n & 1)(t *= a) %= mod;
    return t;
}

int main() {
    int n, m;
    cin >> n >> m;
    std::vector<long long> f = {0, 1};
    for (int i = 2; i < m * 2 + 5; i++)
        f.push_back((f[i - 1] + f[i - 2]) % mod);

    for (auto &t : f) t = Pow(t, m, mod);
    for (int i = 1; i < m * 2 + 5; i++)
        f[i] = (f[i - 1] + f[i]) % mod;
    LinearRecurrence solver(f, mod, false);
    printf("%lld\n", solver.calc(n));
}
```

自适应simpson 积分

```
double simpson(double a,double b) {
    double c = a + (b-a)/2;
    return (F(a) + 4*F(c) + F(b))*(b-a)/6;
}
double asr(double a,double b,double eps,double A) {
    double c = a + (b-a)/2;
    double L = simpson(a,c), R = simpson(c,b);
    if (fabs(L + R - A) <= 15*eps)
        return L + R + (L + R - A)/15.0;
    return asr(a,c,eps/2,L) + asr(c,b,eps/2,R);
}
double asr(double a,double b,double eps) {
    return asr(a,b,eps,simpson(a,b));
}
```

杜教多项式插值

```
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<assert.h>
using namespace std;
typedef long long ll;
const int mod = 1e9+7;
namespace polysum {
    #define rep(i,a,n) for (int i=a;i<n;i++)
    #define per(i,a,n) for (int i=n-1;i>=a;i--)
    const int D=2010;//最高幂次
    ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
    ll powmod(ll a,ll b){ll
res=1;a%=mod;assert(b>=0);for(;b;b>>=1){if(b&1)res=res*a%mod;
a=a*a%mod;}return res;}
    ll calcn(int d,ll *a,ll n) { // a[0].. a[d]  a[n]
        if (n<=d) return a[n];
        p1[0]=p2[0]=1;
        rep(i,0,d+1) {
            ll t=(n-i+mod)%mod;
            p1[i+1]=p1[i]*t%mod;
        }
        rep(i,0,d+1) {
            ll t=(n-d+i+mod)%mod;
            p2[i+1]=p2[i]*t%mod;
        }
        ll ans=0;
        rep(i,0,d+1) {
            ll
t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
            if ((d-i)&1) ans=(ans-t+mod)%mod;
            else ans=(ans+t)%mod;
        }
        return ans;
    }
    void init(int M) {//最高幂次
        f[0]=f[1]=g[0]=g[1]=1;
        rep(i,2,M+5) f[i]=f[i-1]*i%mod;
        g[M+4]=powmod(f[M+4],mod-2);
        per(i,1,M+4) g[i]=g[i+1]*(i+1)%mod;
    }
    ll polysum(ll m,ll *a,ll n) { // a[0].. a[m] \sum_{i=0}^{n-1}
a[i]
        ll b[D];
        for(int i=0;i<=m;i++) b[i]=a[i];
```

```
            b[m+1]=calcn(m,b,m+1);
            rep(i,1,m+2) b[i]=(b[i-1]+b[i])%mod;
            return calcn(m+1,b,n-1);
        }
        ll qpolysum(ll R,ll n,ll *a,ll m) { // a[0].. a[m]
\sum_{i=0}^{n-1} a[i]*R^i
            if (R==1) return polysum(n,a,m);
            a[m+1]=calcn(m,a,m+1);
            ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;
            h[0][0]=0;h[0][1]=1;
            rep(i,1,m+2) {
                h[i][0]=(h[i-1][0]+a[i-1])*r%mod;
                h[i][1]=h[i-1][1]*r%mod;
            }
            rep(i,0,m+2) {
                ll t=g[i]*g[m+1-i]%mod;
                if (i&1)
p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+
mod)%mod;
                else
p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
            }
            c=powmod(p4,mod-2)*(mod-p3)%mod;
            rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]*c)%mod;
            rep(i,0,m+2) C[i]=h[i][0];
            ans=(calcn(m,C,n)*powmod(R,n)-c)%mod;
            if (ans<0) ans+=mod;
            return ans;
        }
    } // polysum::init();
```

求 x^2+y^2=n 的(x,y)对数

```
typedef long long ll;
const ll inf = 1e9+7;
const ll maxn = 2e5+7;
int solve(int n){
    int sum=0;
    for(int i=1;i*i<=n;i++){
        if(n%i==0){
            if(i%4==1)sum++;
            else if(i%4==3)sum--;
            if(i*i!=n){
                if(n/i%4==1)sum++;
                else if(n/i%4==3)sum--;
            }
        }
    }
    return sum*4;
}
int solve2(int n){
    while(n%2==0)n/=2;
    int res=4;
    for(int i=2;i*i<=n;i++){
        if(n%i==0){
            int sum=0;
            while(n%i==0)n/=i,sum++;
            if(i%4==1)
                res=res*(sum+1);
            else if(i%4==3&&sum%2==1)
                return 0;
        }
    }
    if(n>1){
        if(n%4==1)
            res=res*2;
    }
    return res;
}
```