目录

```
#pragma comment(linker, "/STACK:102400000,102400000")
#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>
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--)
typedef unsigned long long ULL;
typedef long long LL;
const int INF=0x3f3f3f3f;
const LL INFF=0x3f3f3f3f3f3f3f3fll;
const LL M=1e9+7;
const LL maxn=1e6+7;
const double eps=0.00000001;
LL gcd(LL a,LL b){return b?gcd(b,a%b):a;}
template<typename T>inline T abs(T a) {return a>0?a:-a;}
template<typename T>inline T powMM(T a,T b){T ret=1;for (;b;b>>=1ll,a=a*a%M) if (b&1) ret=1ll*ret*a%M;return ret;}
```

头文件在上面~~~~~~

杂物

```
int ans;
void fqsort(int l,int r)//第 k 大
{
	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);
}
void msort(int le,int ri)//逆序对
{
	if (le==ri) return;
	int mid=(le+ri)>>1,l1=le,r1=mid+1,k1=l1;
	msort(le,mid); msort(r1,ri);
	while (l1<=mid||r1<=ri)
	{
		if (l1==mid+1) {b[k1++]=a[r1++]; ans+=mid-l1+1;}
		else if (r1==ri+1) b[k1++]=a[l1++];
		else if (a[l1]<=a[r1]) b[k1++]=a[l1++];
		else {b[k1++]=a[r1++]; ans+=mid-l1+1;}
	}
	for (l1=le;l1<=ri;l1++) a[l1]=b[l1];
}
```

输入挂

```
int n,m;
char s[maxn],str[maxn];
int len1,len2,p[maxn],ans;
template<class T>
bool read_d(T &num){
	char in;bool IsN=false;
	in=getchar();
	if (in==EOF) return false;
	while (in!= '-'&&(in<'0'||in>'9')) in=getchar();
	if (in=='-') {IsN=1;num=0;}
	else num=in-'0';
```

```
    while (in=getchar(),in>='0'&&in<='9') num=num*10+in-'0';
    if (IsN) num=-num;
    return 1;
}
template<class T>
bool read_f(T &num){
    char in;bool IsN=false,IsD=false;
    T Dec=0.1;
    in=getchar();
    if (in==EOF) return false;
    while (in!='-'&&in!='.'&&(in<'0'||in>'9')) in=getchar();
    if (in=='-') {IsN=1;num=0;}
    else if (in=='.') {IsD=1;num=0;}
    else num=in-'0';
    if (!IsD)
        while (in=getchar(),in>='0'&&in<='9') num=num*10+in-'0';
    if (in=='.')
        while (in=getchar(),in>='0'&&in<='9') {num+=Dec*(in-'0');Dec*=0.1;}
    if (IsN) num=-num;
    return 1;
}
LL d;
double c;
int main(){
    int i;
    while (read_f(c)){
        printf("%lf\n",c);
    }
}
```

字符串的

KMP

```
LL n,m;
char s[M],a[N];
LL Next[N];
LL i,j,k,t;
void init(char *a,LL *Next){
	Next[0]=-1;
	int len=strlen(a);
	register int i,j;
	FOR(i,1,len-1){
		j=Next[i-1];
		while (j>=0&&a[j+1]!=a[i]) j=Next[j];
		if (a[i]==a[j+1]) Next[i]=j+1;
		else Next[i]=-1;
	}
}
int kmp(char *s,char *a,LL *Next){
	int Len=strlen(s),len=strlen(a);
	register int i,j=-1;
	REP(i,Len){
		while (j>=0&&a[j+1]!=s[i]) j=Next[j];
		if (s[i]==a[j+1]) j++;
		if (j==len-1) return i-len+1;
	}
	return -1;
}
int main(){
	while (~scanf("%s%s",&s,&a)){
		init(a,Next);
		n=strlen(a);
		t=kmp(s,a,Next);
		if (~t) printf("%d",t+1);
		else printf("Not Found!");
		puts("");
	}
}
```

字典树

```
LL n,m;
LL a[N][27],f[N],ff[N];//ff[N]:num
LL i,j,k;
int cnt;
string s;
inline void insert(string str){
	int len=str.length(),now=0;
	int i;
	REP(i,len){
		if (!a[now][str[i]-'a']) a[now][str[i]-'a']=++cnt;
		now=a[now][str[i]-'a'];
		++f[now];//表示小于等于这个的有多少
	}
	ff[now]++;//==的
}
int calc(string str){//小于 str 的
	int len=str.length(),now=0,ans=0;
	int i,j;
	REP(i,len){
		REP(j,str[i]-'a')
			ans+=f[a[now][j]];
//		  if (i!=len-1)//等于的也加
			ans+=ff[a[now][str[i]-'a']];
		now=a[now][str[i]-'a'];
		if (now==0) break;
	}
	return ans;//求大的要再加上后面的
}
int findstr(string str){//等于的
	int len=str.length(),now=0,ans=0,i;
	REP(i,len){
		now=a[now][str[i]-'a'];
		if (now==0) return 0;
	}
	return ans=ff[now];//可能==0
}
int main(){
	scanf("%d%d",&n,&m);
	REP(i,n) {cin>>s;insert(s);}
	REP(i,m) {cin>>s;cout<<calc(s)<<'\n';}
}
//维护 val //left,right 各一个
```

```
//求 i<j<k&&i^j<j^k 的三元组个数
int T;
int n;
LL ans;
int i,j;
int a[maxn*32];
int nxt[maxn*32][2];
LL num[maxn*32],last[maxn*32];
LL sum[maxn][32][2];//只有这位。。。
int cnt,now;
int main()
{
    scanf("%d",&T);
    while (T--){
        ans=0;
        cnt=0;
        scanf("%d",&n);
        FOR(i,1,n) scanf("%d",&a[i]);
        FOR(i,1,n){
            rREP(j,32){
                int mark=((a[i]&(1<<j))!=0);
                sum[i][j][0]=sum[i-1][j][0];
                sum[i][j][1]=sum[i-1][j][1];
                sum[i][j][mark]++;
            }
        }
        rFOR(i,1,n){
            now=0;
            rREP(j,32){
                int mark=((a[i]&(1<<j))!=0);
                if (nxt[now][mark^1])
                    ans+=last[nxt[now][mark^1]]-
num[nxt[now][mark^1]]*sum[i][j][mark];
                if (!nxt[now][mark]) break;
                now=nxt[now][mark];
            }
            now=0;
            rREP(j,32){
                int mark=((a[i]&(1<<j))!=0);
                if (!nxt[now][mark]) nxt[now][mark]=++cnt;
                now=nxt[now][mark];
                last[now]+=sum[i-1][j][mark^1];//这点之前
                num[now]++;
            }
```

```
            }
            printf("%lld\n",ans);
            FOR(i,0,cnt) num[i]=last[i]=nxt[i][0]=nxt[i][1]=0;
            FOR(i,1,n)
                REP(j,32) sum[i][j][0]=sum[i][j][1]=0;
        }
}
```

AC 自动机

//HDU2222 多串在一个串内出现次数

```
const int maxtot=50*10007;//个数
const int charnum=26;
int nxt[maxtot][charnum],fail[maxtot],num[maxtot];
int cnt;
queue<int> Q;
void init(){
    int i,j;
    while (Q.size()) Q.pop();
    REP(i,maxtot) {
        REP(j,charnum) nxt[i][j]=0;
        num[i]=fail[i]=0;
    }
    cnt=1;
}
inline void insert(char *str){
    int len=strlen(str),now=0,i;
    REP(i,len){
        int k=str[i]-'a';
        if (!nxt[now][k]) nxt[now][k]=cnt++;
        now=nxt[now][k];
    }
    num[now]++;
}
inline void buildAC(){
    fail[0]=-1;
    Q.push(0);
    int i;
    while (Q.size()){
        int x=Q.front();Q.pop();
        REP(i,charnum) if (nxt[x][i]){
            if (x==0) fail[nxt[x][i]]=0;
            else {
                int p=fail[x];
```

```
                while (p!=-1&&!nxt[p][i]) p=fail[p];//注意这里是 nxt[p][i]
                if (p!=-1) fail[nxt[x][i]]=nxt[p][i];
                else fail[nxt[x][i]]=0;
            }
            Q.push(nxt[x][i]);
        }
    }
}
inline int match(char *str){
    int len=strlen(str),now=0;
    int i,ret=0;
    REP(i,len){
        int k=str[i]-'a';
        while (now&&!nxt[now][k]) now=fail[now];
        now=nxt[now][k];
        if (now==-1) now=0;
        int tmp=now;
        while (tmp){
            if (num[tmp]==-1) break;//vis
            ret+=num[tmp];
            num[tmp]=-1;
            tmp=fail[tmp];
        }
    }
    return ret;
}
int T,i,n;
char s[maxn];
int main(){
    scanf("%d",&T);
    while (T--){
        scanf("%d",&n);
        init();
        REP(i,n){
            scanf("%s",s);
            insert(s);
        }
        buildAC();
        scanf("%s",s);
        printf("%d\n",match(s));
    }
}
```

//HDU2896 输出串

```
int ans[505],num;//标记
const int tot=505; const int maxtot=505*140; const int charnum=98;
int nxt[maxtot][charnum],fail[maxtot],mark[maxtot];
int cnt;
queue<int> Q;
void init(){
	int i,j;
	while (Q.size()) Q.pop();
	REP(i,maxtot){
		REP(j,charnum) nxt[i][j]=0;
		mark[i]=fail[i]=0;
	}
	cnt=1;
}
inline void insert(char *str,int id){
	int len=strlen(str),now=0,i;
	REP(i,len){
		int k=str[i]-33;
		if (!nxt[now][k]) nxt[now][k]=cnt++;
		now=nxt[now][k];
	}
	mark[now]=id;
}
inline void buildAC(){
	fail[0]=-1;
	Q.push(0);
	int i;
	while (!Q.empty()){
		int x=Q.front();Q.pop();
		REP(i,charnum) if (nxt[x][i]){
			if (x==0) fail[nxt[x][i]]=0;
			else{
				int p=fail[x];
				while (p!=-1&&!nxt[p][i]) p=fail[p];//这里注意
				if (p!=-1) fail[nxt[x][i]]=nxt[p][i];
				else fail[nxt[x][i]]=0;
			}
			Q.push(nxt[x][i]);
		}
	}
}
inline void match(char *str){
	int len=strlen(str),now=0;
	int i;
```

```
        num=0;
        REP(i,tot) ans[i]=0;
        REP(i,len){
            int k=str[i]-33;
            while (now&&!nxt[now][k]) now=fail[now];
            now=nxt[now][k];
            if (now==-1) now=0;
            int tmp=now;
            while (tmp&&!ans[mark[tmp]]){
                if (mark[tmp]){
                    ans[mark[tmp]]=1;
                    num++;
                }
                tmp=fail[tmp];
                if (num>=3) return;
            }
        }
}
int T,i,j,n,m,total;
char s[maxn];
int main(){
        while (~scanf("%d",&n)){
            total=0;
            init();
            REP(i,n){
                scanf("%s",s);
                insert(s,i+1);
            }
            buildAC();
            scanf("%d",&m);
            REP(i,m){
                scanf("%s",s);
                match(s);
                if (num==0) continue;
                total++;
                printf("web %d:",i+1);
                REP(j,tot) if (ans[j]) printf(" %d",j);
                puts("");
            }
            printf("total: %d\n",total);
        }
}
```

后缀数组

HDU6138,前缀+公共子串

```
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;
    }
}
int n,m;
int i,j,k;
char a[maxn];
int s[maxn];
```

```
int st[maxn];
int sa[maxn];
int id[maxn];
int val[maxn];
int tot,now,ans;
int main(){
    int T;
    scanf("%d",&T);
    while (T--){
        scanf("%d",&n);
        tot=0;
        FOR(i,1,n){
            scanf("%s",a);
            int len=strlen(a);
            st[tot]=len;
            REP(j,len) id[tot]=i,s[tot++]=a[j]-'a'+1;
            s[tot++]='z'-'a'+i+1;
        }
        s[tot]=0;
        da(s,sa,tot,26+n+1);
        calheight(s,sa,tot);
        now=0;
        FOR(i,1,tot){
            val[i]=max(val[i],now);
            now=min(now,height[i+1]);
            if (st[sa[i]]) now=max(now,height[i+1]),val[i]=INF;
        }
        now=0;//这里可以改成三个标记取 min
        rFOR(i,1,tot){
            val[i]=max(val[i],now);
            now=min(now,height[i]);
            if (st[sa[i]]) now=max(now,height[i]),val[i]=max(val[i],st[sa[i]]);
        }
        char S[maxn];
        REP(i,tot) S[i]=s[i]+'a'-1;S[tot]=0;
        scanf("%d",&m);
        REP(i,m){
            int x,y,i;
            scanf("%d%d",&x,&y);
            now=0;
            ans=0;
            FOR(i,1,tot){
                if (id[sa[i]]==x&&st[sa[i]]) now=max(now,st[sa[i]]);
                if (id[sa[i]]==y) ans=max(ans,min(now,val[i]));
```

```
                now=min(now,height[i+1]);
                if (id[sa[i]]==x) now=max(now,height[i+1]);
            }
            now=0;
            rFOR(i,1,tot){
                if (id[sa[i]]==x&&st[sa[i]]) now=max(now,st[sa[i]]);
                if (id[sa[i]]==y) ans=max(ans,min(now,val[i]));
                now=min(now,height[i]);
                if (id[sa[i]]==x) now=max(now,height[i]);
            }
            printf("%d\n",ans);
        }
        FOR(i,1,tot) val[i]=st[i]=0;
    }
}
```

后缀自动机

```
int nxt[maxn][26],pre[maxn],len[maxn];
```

//pre 为上一个可以接受的位置 (树形结构,前缀相等,类似 AC 自动机 fail 指针),这样可以去除很多无用的边

//注意,pre 的边不是所有的边!所以反过来求 num 的时候不能直接用 pre,要 REP(i,26),但是最长公共子串是要 pre 的

```
int cnt,last;
void add(int c){
    int np=++cnt,p=last;
    len[np]=len[p]+1;
    for (;p&&!nxt[p][c];p=pre[p]) nxt[p][c]=np;//边表示字符
    if (!p) pre[np]=1;
    else{
        int q=nxt[p][c];
        if (len[p]+1==len[q]) pre[np]=q;
        else{
            int nq=++cnt;len[nq]=len[p]+1;//new 一个新节点(松弛(copy 一遍))来保证结构稳定(或 len 相等)
            memcpy(nxt[nq],nxt[q],sizeof(nxt[q]));
            pre[nq]=pre[q];
            pre[np]=pre[q]=nq;
            for (;p&&nxt[p][c]==q;p=pre[p]) nxt[p][c]=nq;
        }
    }
    last=np;
}
//void dfs(int x,int len){
```

```
//    int i;
//    printf("%s\n",a);
//    REP(i,27){
//        if (nxt[x][i]){
//            a[len]=i+'a';
//            dfs(nxt[x][i],len+1);
//            a[len]=0;
//        }
//    }
//}
char a[maxn],b[maxn];
int F[maxn][10];
int n,m;
int i,j,k;
int ans,now,nowlen;
int T;
int S[maxn],K[maxn];
int main()
{
    scanf("%s",a);
    n=strlen(a);
    last=++cnt;//1 开始
    REP(i,n) add(a[i]-'a');
    T=0;
    while (~scanf("%s",&b)){
        T++;
        m=strlen(b);
        now=1;nowlen=0;
        REP(i,m){
            while (now&&!nxt[now][b[i]-'a']) now=pre[now],nowlen=len[now];
            if (!now) now=1,nowlen=0;
            if (nxt[now][b[i]-'a']){
                now=nxt[now][b[i]-'a'];
                nowlen++;
            }
            F[now][T]=max(nowlen,F[now][T]);
        }
    }
    FOR(i,1,cnt) S[len[i]]++;
    FOR(i,1,n) S[i]+=S[i-1];
    FOR(i,1,cnt) K[S[len[i]]--]=i;
    rFOR(i,1,cnt){
        FOR(j,1,T) F[pre[K[i]]][j]=max(F[pre[K[i]]][j],min(F[K[i]][j],len[pre[K[i]]]));
    }
```

```
    FOR(i,1,cnt){
        int mx=INF;
        FOR(j,1,T) mx=min(mx,F[i][j]);
        ans=max(ans,mx);
    }
    printf("%d\n",ans);
}
```

后缀自动机+主席树合并

//查询某串部分在串 l->r 的最大出现次数及位置

```
//SAM
int nxt[maxn][27],pre[maxn],len[maxn];
int CNT,last;
void add(int c){
    int np=++CNT,p=last;
    len[np]=len[p]+1;
    for (;p&&!nxt[p][c];p=pre[p]) nxt[p][c]=np;
    if (!p) pre[np]=1;
    else{
        int q=nxt[p][c];
        if (len[p]+1==len[q]) pre[np]=q;
        else{
            int nq=++CNT;len[nq]=len[p]+1;
            memcpy(nxt[nq],nxt[q],sizeof(nxt[q]));
            pre[nq]=pre[q];
            pre[np]=pre[q]=nq;
            for (;p&&nxt[p][c]==q;p=pre[p]) nxt[p][c]=nq;
        }
    }
    last=np;
}
//char A[maxn];
//void dfs(int x,int len){//check
//    int i;
//    printf("%s\n",A);
//    REP(i,26){
//        if (nxt[x][i]){
//            A[len]=i+'a';
//            dfs(nxt[x][i],len+1);
//            A[len]=0;
//        }
//    }
//}
```

```
//segtree
int cnt;
struct node{
	pair<int,int> val;//bigger
	int l,r;
}tree[maxn*25];
int root[maxn];
inline pair<int,int> add(pair<int,int> A,pair<int,int> B){
	return make_pair(A.first+B.first,A.second);
}
inline pair<int,int> better(pair<int,int> A,pair<int,int> B){
	if (A.first==B.first) return A.second<B.second?A:B;
	return A.first>B.first?A:B;
}
inline void insert(int &x,int val,int l,int r){
	if (!x) x=++cnt;
	if (l==r){
		tree[x].val.first++;
		tree[x].val.second=l;
		return;
	}
	int mid=(l+r)/2;
	if (val<=mid) insert(tree[x].l,val,l,mid);
	else insert(tree[x].r,val,mid+1,r);
	tree[x].val=better(tree[tree[x].l].val,tree[tree[x].r].val);
}
inline int Merge(int x,int y,int l,int r){
	if (!x||!y) return x|y;
	int z=++cnt;
	if (l==r){
		tree[z].val=add(tree[x].val,tree[y].val);
		return z;
	}
	int mid=(l+r)/2;
	tree[z].l=Merge(tree[x].l,tree[y].l,l,mid);
	tree[z].r=Merge(tree[x].r,tree[y].r,mid+1,r);
	tree[z].val=better(tree[tree[z].l].val,tree[tree[z].r].val);
	return z;
}
inline pair<int,int> query(int x,int l,int r,int L,int R){
	if (!x) return make_pair(0,0);
	if (l<=L&&R<=r) return tree[x].val;
	int mid=(L+R)/2;
	pair<int,int> ret=make_pair(0,0);
```

```
    if (mid>=l) ret=better(ret,query(tree[x].l,l,r,L,mid));
    if (r>mid) ret=better(ret,query(tree[x].r,l,r,mid+1,R));
    return ret;
}
int father[21][maxn],pos[maxn];//倍增求 father
inline int getfather(int l,int r){
    int L=(r-l+1),ret=pos[r],i;
    rFOR(i,0,20) if (len[father[i][ret]]>=L) ret=father[i][ret];
    return ret;
}
int n,m,q;
int i,j,k;
char s[maxn];
int S[maxn],K[maxn];
int main(){
    scanf("%s",s);
    last=++CNT;
    n=strlen(s);
    REP(i,n) add(s[i]-'a'),pos[i+1]=last;
    add(26);
    scanf("%d",&m);
    FOR(k,1,m){
        scanf("%s",s);
        n=strlen(s);
        REP(i,n) add(s[i]-'a'),insert(root[last],k,1,m);
        add(26);
    }
    FOR(i,1,CNT) S[len[i]]++;
    FOR(i,1,CNT) S[i]+=S[i-1];
    FOR(i,1,CNT) K[S[len[i]]--]=i;
    rFOR(i,1,CNT){
        if (pre[K[i]]) root[pre[K[i]]]=Merge(root[pre[K[i]]],root[K[i]],1,m);
    }
    FOR(i,1,CNT) father[0][i]=pre[i];
    FOR(j,1,20)
        FOR(i,1,CNT) father[j][i]=father[j-1][father[j-1][i]];//倍增
    scanf("%d",&q);
    while (q--){
        int l,r,pl,pr;
        scanf("%d%d%d%d",&l,&r,&pl,&pr);
        int x=getfather(pl,pr);
        pair<int,int> ans=query(root[x],l,r,1,m);
        if (ans.first==0) printf("%d 0\n",l);
        else printf("%d %d\n",ans.second,ans.first);
```

```
    }
}
```

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

```
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;
//  printf("%s",str);
}
void manacher(){//主要是说已经对称匹配过的不用再进行
    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;
        }
    }
}
```

//滚动的最长回文子串

```
int a[maxn];
struct node{
    int left,right;
}tree[maxn*4*8];
int val[maxn*4*8],lazy[maxn*4*8];
void change(int x,int i){
    val[x]=max(val[x],i);
    lazy[x]=max(lazy[x],i);
}
```

```
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){
	tree[x].left=l;tree[x].right=r;
	val[x]=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 update(int x,int l,int r,LL val){
	int L=tree[x].left,R=tree[x].right;
	if (l<=L&&R<=r){
		change(x,val);
		return;
	}
	pushdown(x);
	int mid=(L+R)/2;
	if (mid>=l) update(x<<1,l,r,val);
	if (r>mid) update(x<<1|1,l,r,val);
}
int query(int x,int pos){
	int L=tree[x].left,R=tree[x].right;
	if (L==R) return val[x];
	pushdown(x);
	int mid=(L+R)/2;
	if (mid>=pos) return query(x<<1,pos);
	return query(x<<1|1,pos);
}
int n,m;
char s[maxn*2],str[maxn*4];
int len1,len2,p[maxn*8];
//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
int i,j,k;
int del1[maxn*8],del2[maxn*8];
int ans[maxn*8];
int main(){
	scanf("%d",&n);
	scanf("%s",s);
```

```
    rep(i,n,n*2) s[i]=s[i-n];
//init();
    int i;
    len1=strlen(s);
    str[0]='+';str[1]='%';
    REP(i,len1+1){
        str[i*2+2]=s[i];
        str[i*2+3]='%';
    }
    len2=len1*2+2;
//manacher();
    int id=0,mx=0;
    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;
        }
    }
    REP(i,len2) p[i]--;//manacher
//solve
    REP(i,len2) {
        if ((p[i]&1)==(n&1)) p[i]=min(p[i],n);
        else p[i]=min(p[i],n-1);
    }
    build(1,1,len2*2);
    REP(i,len2){
        del1[i-p[i]]=max(del1[i-p[i]],p[i]);
        if (i+p[i]-n*2>=0) del2[i+p[i]-n*2]=max(del2[i+p[i]-n*2],p[i]);
        if (i+p[i]-n*2<i-p[i]&&i-p[i]>0){
            update(1,max(0,i+p[i]-n*2)+1,max(0,i-p[i])+1,p[i]);
        }
    }
    mx=0;
    REP(i,len2){
        if (str[i]!='%'&&str[i]!='+') mx-=2;
        mx=max(mx,del1[i]);
        ans[i]=max(ans[i],mx);
    }
    mx=0;
    rREP(i,len2*2){
        if (str[i]!='%'&&str[i]!='+') mx-=2;
```

```
            mx=max(mx,del2[i]);
            ans[i]=max(ans[i],mx);
        }
        REP(i,len2) ans[i]=max(ans[i],query(1,i+1));
        REP(i,n) printf("%d\n",max(ans[i*2+1],ans[i*2+2]));
}
```

回文自动机

//两串相同回文子串

```
struct Ptree{
        int next[maxn][27];
        int fail[maxn];
        int cnt[maxn];//真正个数
//      int num[maxn];//右端点结尾的 maxnum
        int len[maxn];//长度
        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]));
                cnt[tot]=0;
//              num[tot]=0;
                len[tot]=l;//不是 1...
                return tot++;
        }
        void init(){
                tot=0;
                newnode(0);
                newnode(-1);
                last=n=0;
                S[n]=-1;//减少特判
                fail[0]=1;
        }
        int getfail(int x){
                while(S[n-len[x]-1]!=S[n]) x=fail[x];
                return x;
        }
        void add(int c){
                c-='a';
                S[++n]=c;
                int cur=getfail(last);
                if (!next[cur][c]){
                        int now=newnode(len[cur]+2);
```

```
                fail[now]=next[getfail(fail[cur])][c];
                next[cur][c]=now;
//              num[now]=num[fail[now]]+1;
            }
            last=next[cur][c];
            cnt[last]++;
        }
        void count(){//count 完才对
            int i;
            rREP(i,tot) cnt[fail[i]]+=cnt[i];
        }
}T1,T2;
LL ans;
void dfs(int x,int y){
        int i;
        REP(i,27){
            int u=T1.next[x][i],v=T2.next[y][i];
            if (u&&v){
                ans+=1ll*T1.cnt[u]*T2.cnt[v];
                dfs(u,v);
            }
        }
}
char a[maxn],b[maxn];
void solve(){
        scanf("%s%s",a,b);
        int len1=strlen(a),len2=strlen(b);
        int i,j;
        T1.init();T2.init();
        REP(i,len1) T1.add(a[i]);
        REP(j,len2) T2.add(b[j]);
        T1.count();
        T2.count();
        dfs(0,0);
        dfs(1,1);
}
int main(){
        int T,x=0;
        scanf("%d",&T);
        while (T--) {
            ans=0;
            solve();
            printf("Case #%d: %lld\n",++x,ans);
        }
```

板子？？？

```
}
```

数据结构

按秩合并并查集(+整体二分)

```
//求删去每个点后图是否存在奇环(主要是整体二分思想)
typedef pair<int,int>    pii;
#define fi first
#define se second
#define mp make_pair
vector<pii> E[maxn<<2],have[maxn<<2],back[maxn<<2];//防爆栈
int fa[maxn],val[maxn];
pii getfa(int x){
    int ret=x,color=val[ret];
    while (fa[ret]!=ret) ret=fa[ret],color^=val[ret];
    return mp(ret,color);
}
int sz[maxn];
int ans[maxn];
void solve(int X,int l,int r){
    bool flag=0;
    int i;
    int mid=(l+r)/2;
    for(pii e:have[X]){
        pii x=getfa(e.fi);
        pii y=getfa(e.se);
        if (x.fi==y.fi){
            if (x.se==y.se){
                flag=1;
                break;
            }
        }else{
            if (sz[x.fi]>sz[y.fi]) swap(x,y);
            back[X].push_back(mp(x.fi,x.se^y.se));
            fa[x.fi]=y.fi;
            sz[y.fi]+=sz[x.fi];
            val[x.fi]^=x.se^y.se;
        }
    }
    if (flag){
        FOR(i,l,r) ans[i]=0;
    }else if (l<r){
        int mid=(l+r)/2;
```

```
            for (pii e:E[X]){
                if ((l<=e.fi&&e.fi<=mid)||(l<=e.se&&e.se<=mid)) E[X<<1].push_back(e);
                else have[X<<1].push_back(e);
                if ((mid+1<=e.fi&&e.fi<=r)||(mid+1<=e.se&&e.se<=r))
E[X<<1|1].push_back(e);
                else have[X<<1|1].push_back(e);
            }
            solve(X<<1,l,mid);
            solve(X<<1|1,mid+1,r);
        }
        for (pii u:back[X]){
            sz[fa[u.fi]]-=sz[u.fi];
            fa[u.fi]=u.fi;
            val[u.fi]^=u.se;
        }
        vector<pii>().swap(E[X]);
        vector<pii>().swap(have[X]);
        vector<pii>().swap(back[X]);
    }
    int n,m;
    int i;
    int main()
    {
        int T;
        scanf("%d",&T);
        while (T--){
            scanf("%d%d",&n,&m);
            FOR(i,1,n) fa[i]=i,sz[i]=1,ans[i]=1,val[i]=1;
            FOR(i,1,m){
                int u,v;
                scanf("%d%d",&u,&v);
                if (u>v) swap(u,v);
                E[1].push_back(make_pair(u,v));
            }
            solve(1,1,n);
            FOR(i,1,n) printf("%d",ans[i]);puts("");
        }
    }
```

树状数组

/*区间最大值*/

```
LL a[N];int n;int i,j,k;
LL lowbit(LL x){return x&-x;}
```

```
LL m[N];
void change(LL r){
	m[r]=a[r];
	LL i,t=lowbit(r);
	for (i=1;i<t;i<<=1) m[r]=max(m[r],m[r-i]);
}
void init(LL n){
	LL i;
	FOR(i,1,n) c[i]=0;
	FOR(i,1,n) change(i);
}
void update(LL x){
	LL i;
	change(x);
	for (i=x;i<=n;i+=lowbit(i)) change(i);
}
LL getmax(LL l,LL r){
	LL ret=a[r];
	while (l!=r){
		for (r--;r-lowbit(r)>=l;r-=lowbit(r)) ret=max(ret,m[r]);
		ret=max(ret,a[r]);
	}
	return ret;
}
int main()
{
	cin>>n;
	FOR(i,1,n) cin>>a[i];
	init(n);
	FOR(i,1,n) cout<<m[i]<<' ';
	cin>>n;
	FOR(i,1,n){
		cin>>j>>k;
		printf("%lld\n",getmax(j,k));
	}
}
```

/*区间和，单点修改*/

```
LL a[N];
int n,m;
int i,j,k;
LL lowbit(LL x){
	return x&-x;
}
```

```
LL c[N];
LL presum(LL x){
	LL ret=0;
	while (x){
		ret+=c[x];
		x-=lowbit(x);//可^=
	}
	return ret;
}
LL sum(LL l,LL r){
	return presum(r)-presum(l-1);
}
void add(LL x,int d){//修改不如 add 有效
	while (x<=n){
		c[x]+=d;
		x+=lowbit(x);
	}
}
void init(LL n){
	FOR(i,1,n) c[i]=0;
	FOR(i,1,n) add(i,a[i]);
}
int main()
{
	cin>>n;
	FOR(i,1,n) cin>>a[i];
	init(n);
	FOR(i,1,n) cout<<c[i]<<' ';
	cin>>n;
	FOR(i,1,n){
		cin>>j>>k;
		printf("%lld\n",sum(j,k));
	}
}
```

二维树状数组

//区间修改单点查询(仅 0|1)

```
int n,m;
int c[maxn][maxn];
int lowbit(int x){return x&-x;}
void update(int x1,int y1){
	int x=x1;
	while (x<=n){
```

```
            int y=y1;
            while (y<=n){
                c[x][y]^=1;
                y+=lowbit(y);
            }
            x+=lowbit(x);
        }
}
int sum(int x1,int y1){
        int ret=0;
        int x=x1;
        while (x){
            int y=y1;
            while (y){
                ret^=c[x][y];
                y^=lowbit(y);
            }
            x^=lowbit(x);
        }
        return ret;
}
void init(){
        int i,j;
        FOR(i,1,n)
            FOR(j,1,n) c[i][j]=0;
}
int T;
char s[10];
int i,j,k;
int x1,x2,y1,y2;
int main()
{
        scanf("%d",&T);
        while (T--){
            scanf("%d%d",&n,&m);
            init();
            REP(i,m){
                scanf("%s",s);
                if (s[0]=='C'){
                    scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
                    update(x1,y1);
                    update(x1,y2+1);
                    update(x2+1,y1);
                    update(x2+1,y2+1);
```

```
                }
                else {
                    scanf("%d%d",&x1,&y1);
                    printf("%d\n",sum(x1,y1));
                }
            }
            puts("");
        }
}
```

树状数组 不大于 k 的最大值

```
int a[maxn];
int n,i,j;
const int nn=1000000;
inline int lowbit(int x){
    return x&-x;
}
inline void insert(int x){
    while (x<=nn){
        a[x]++;
        x+=lowbit(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;
}
int ans;
const int MOD=19260817;
int main()
{
    while(~scanf("%d",&n))
    {
        ans=0;
        FOR(i,1,1000000) a[i]=0;
        REP(i,n){
```

```
                scanf("%d",&j);
                if (j==0) continue;
                ans=ans+find(j);
//              printf("%d ",find(j));
                insert(j);
                ans%=MOD;
            }
            printf("%d\n",ans);
        }
}
```

线段树

//我自己感觉写的恶心的要死..不好看

```
int a[maxn];
LL sum[maxn<<2],lazy[maxn<<2],len[maxn<<2],mx[maxn<<2],mn[maxn<<2];
int lef[maxn<<2],rig[maxn<<2];
void change(int x,LL val){
	sum[x]=len[x]*val;
	mx[x]=mn[x]=val;
	lazy[x]=val;
}
void pushdown(int x){
	if (lazy[x]!=-1){
		change(x<<1,lazy[x]);
		change(x<<1|1,lazy[x]);
		lazy[x]=-1;
	}
}
void pushup(int x){
	sum[x]=sum[x<<1]+sum[x<<1|1];
	len[x]=len[x<<1]+len[x<<1|1];
	mx[x]=max(mx[x<<1],mx[x<<1|1]);
	mn[x]=min(mn[x<<1],mn[x<<1|1]);
}
void build(int x,int l,int r){
	lef[x]=l;rig[x]=r;
	sum[x]=len[x]=0;lazy[x]=-1;
	if (l==r){
		sum[x]=mx[x]=mn[x]=a[l];
		len[x]=1;
		return;
	}
	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,int L,int R,LL val){
	if (mx[x]<val) return;
	if (l<=L&&R<=r&&mn[x]>=val){
		change(x,val);
		return;
	}
	if (L==R) return;
```

```
    pushdown(x);
    int mid=(L+R)/2;
    if (mid>=l) update(x<<1,l,r,L,mid,val);
    if (r>mid) update(x<<1|1,l,r,mid+1,R,val);
    pushup(x);
}
LL query(int x,int l,int r,int L,int R){
    if (l<=L&&R<=r) return sum[x];
    pushdown(x);
    int mid=(L+R)/2;
    LL ret=0;
    if (mid>=l) ret+=query(x<<1,l,r,L,mid);
    if (r>mid) ret+=query(x<<1|1,l,r,mid+1,R);
    pushup(x);
    return ret;
}
```

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),yl=max(1,y-r),yr=min(n,y+r);
            int MX=querymaxX(1,xl,xr,yl,yr),MN=queryminX(1,xl,xr,yl,yr);
            updateX(1,x,y,(MX+MN)/2);
            printf("%d\n",(MX+MN)/2);
        }
    }
}
```

扫描线 矩形周长并

```
int size;
int len[maxn*2];
int n,m;
int 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()
{
    int TT=0;
    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);
}
int n,m;
int i,j,k,ii;
int main()
{
	scanf("%d%d",&n,&m);
	FOR(i,1,n) scanf("%d",&a[i]),v.push_back(a[i]);
	sort(v.begin(),v.end());v.erase(unique(v.begin(),v.end()),v.end());
	FOR(i,1,n) update(1,n,root[i],root[i-1],getid(a[i]));
	REP(ii,m){
		scanf("%d%d%d",&i,&j,&k);
		printf("%d\n",v[query(1,n,root[i-1],root[j],k)-1]);
	}
	return 0;
}
```

区间不重复数字个数和第 k 个是哪位

```
int cnt;
struct node{
	int l,r,sum;
}T[maxn*40];
```

```
void update(int l,int r,int &x,int y,int pos,int v){
	T[++cnt]=T[y],T[cnt].sum+=v,x=cnt;
	if (l==r) return;
	int mid=(l+r)/2;
	if (mid>=pos) update(l,mid,T[x].l,T[y].l,pos,v);
	else update(mid+1,r,T[x].r,T[y].r,pos,v);
}
int findsum(int l,int r,int x,int L,int R){
//每个点记录的都是这个点往后的相同数(前面把后面短路了)
	if (L<=l&&r<=R) return T[x].sum;
	int mid=(l+r)/2;
	int sum=0;
	if (mid>=L) sum+=findsum(l,mid,T[x].l,L,R);
	if (R>mid) sum+=findsum(mid+1,r,T[x].r,L,R);
	return sum;
}
int query(int l,int r,int x,int k){
	if (l==r) return l;
	int mid=(l+r)/2;
	int sum=T[T[x].l].sum;
	if (sum>=k) return query(l,mid,T[x].l,k);
	else return query(mid+1,r,T[x].r,k-sum);
}
int n,m;
int i,j,k,pos;
int t,TT;
int ans[maxn],a[maxn];
int last[maxn],root[maxn];
int main()
{
	scanf("%d",&TT);
	FOR(t,1,TT){
		scanf("%d%d",&n,&m);
		FOR(i,1,n) scanf("%d",&a[i]);
		FOR(i,1,n) last[a[i]]=0,root[i]=0;
		cnt=0;
		rFOR(i,1,n){
			if (!last[a[i]]) update(1,n,root[i],root[i+1],i,1);
			else {
				update(1,n,root[i],root[i+1],last[a[i]],-1);
				update(1,n,root[i],root[i],i,1);
			}
			last[a[i]]=i;
		}
```

```
            FOR(i,1,m){
                scanf("%d%d",&j,&k);
                j=(j+ans[i-1])%n+1;
                k=(k+ans[i-1])%n+1;
                if (j>k) swap(j,k);
                pos=(findsum(1,n,root[j],j,k)+1)/2;
                ans[i]=query(1,n,root[j],pos);
            }
            printf("Case #%d:",t);
            FOR(i,1,m) printf(" %d",ans[i]);
            puts("");
        }
        return 0;
}
```

可持久化数组

```
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;
		}
	}
	return 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);
}
int main(){
	T[0].cnt=T[0].l=T[0].r=0;
	int T_T;
	scanf("%d",&T_T);
	while (T_T--) solve();
}
```

SPLAY

```
int a[maxn],cnt;
struct splay_tree{
	struct node{
		int val,min,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=rev=son[0]=son[1]=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);
        T[tot].init(val);
        fa[tot]=v;
```

```
        T[v].son[0]=tot++;
        pushup(v);pushup(u);
    }
    void erase(int x){
        int u=select(x-1),v=select(x+1);
        splay(u,0);
        splay(v,u);
        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 dfs(int x,int k){//小于 k 的值个数,会被卡
        if (x==0) return 0;
        if (T[x].min!=INF&&T[x].min>=k) return 0;
        if (T[x].max!=-INF&&T[x].max<k) return T[x].size;
        int ret=T[x].val<k;
        if (T[x].son[0]) ret+=dfs(T[x].son[0],k);
        if (T[x].son[1]) ret+=dfs(T[x].son[1],k);
        return ret;
    }
//小于 k 的值个数,会被卡 应该套主席树(但是太长，两个 log)
    int query(int l,int r,int k){
        int u=select(l-1),v=select(r+1);
        splay(u,0);splay(v,u);
        return dfs(T[v].son[0],k);
    }
    int build(int l,int r){
```

```
        if (l>r) return 0;
        if (l==r) return l;
        int mid=(l+r)/2;
        T[mid].son[0]=build(l,mid-1);
        T[mid].son[1]=build(mid+1,r);
        fa[T[mid].son[0]]=fa[T[mid].son[1]]=mid;
        pushup(mid);
        return mid;
    }
    void init(int n){
        tot=0;
        int i;//0 是虚的;
        T[tot++].init(INF);//空的
        T[tot++].init(INF);//前后两个-INF 节点
        FOR(i,1,n) T[tot++].init(a[i]);
        T[tot++].init(INF);
        root=build(1,tot-1);
        fa[root]=0;
        fa[0]=0;T[0].son[1]=root;T[0].size=0;
    }
    void print(int now=-1){
        if (now==-1) now=root;
        pushdown(now);
        if (T[now].son[0]) print(T[now].son[0]);
        if (T[now].val!=-INF){
            if (cnt++) printf(" ");
            printf("%d",T[now].val);
        }
        if (T[now].son[1]) print(T[now].son[1]);
        pushup(now);
    }
}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;
    int n,m;
    vector<int> edge[maxn];
    LL ans[maxn];
    int val[maxn];
    void dfs(int x,int fa){
        ans[x]=val[x];
        for (int v:edge[x]){
            if (v==fa) continue;
            dfs(v,x);
```

```
            ans[x]=T.merge(x,v,ans[x],ans[v]);
        }
}
int i,j,k;
int main(){
        int TT;
        scanf("%d",&TT);
        while (TT--){
            scanf("%d",&n);
            FOR(i,1,n) scanf("%d",&val[i]);
            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) T.T[i].init(val[i]);
            dfs(1,0);
            FOR(i,1,n) printf("%lld ",ans[i]);
            puts("");
            FOR(i,1,n) T.fa[i]=0;
            FOR(i,1,n) ans[i]=0,vector<int>().swap(edge[i]);
        }
}
```

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

```
struct LCT{
        struct node{
            int son[2],val,size;
            int max,add,cnt1;//max
            int ans,lazy,cnt2;//second
            bool rev;
            void init(int _val){
                son[0]=son[1]=rev=add=0;
                max=val=_val;
                size=1;
                cnt1=1;cnt2=0;
                ans=lazy=-INF;
            }
        }T[maxn];
        bool root[maxn];
```

```
int fa[maxn];
void Reverse(int x){
	T[x].rev^=1;
	swap(T[x].son[0],T[x].son[1]);
}
void Add(int x,int val){
	T[x].max+=val;
	T[x].add+=val;
	T[x].val+=val;
	if (T[x].ans!=-INF) T[x].ans+=val;;
	if (T[x].lazy!=-INF) T[x].lazy+=val;
}
void Change(int x,int val){//先 change
	T[x].max=val;
	T[x].add=0;
	T[x].val=val;
	T[x].ans=-INF;
	T[x].cnt2=-INF;
	T[x].cnt1=T[x].size;
	T[x].lazy=val;
}
void Update(int x,int val,int num){
	if (T[x].max==val) T[x].cnt1+=num;
	else if (T[x].max<val){
		T[x].ans=T[x].max;
		T[x].cnt2=T[x].cnt1;
		T[x].max=val;
		T[x].cnt1=num;
	}
	else if (T[x].ans==val) T[x].cnt2+=num;
	else if (T[x].ans<val){
		T[x].ans=val;
		T[x].cnt2=num;
	}
}
void pushup(int x){
	T[x].size=1;
	T[x].max=T[x].val;
	T[x].ans=T[x].lazy=-INF;
	T[x].cnt1=1;T[x].cnt2=0;
	if (T[x].son[0]){
		Update(x,T[T[x].son[0]].max,T[T[x].son[0]].cnt1);
		Update(x,T[T[x].son[0]].ans,T[T[x].son[0]].cnt2);
		T[x].size+=T[T[x].son[0]].size;
```

```
        }
        if (T[x].son[1]){
            Update(x,T[T[x].son[1]].max,T[T[x].son[1]].cnt1);
            Update(x,T[T[x].son[1]].ans,T[T[x].son[1]].cnt2);
            T[x].size+=T[T[x].son[1]].size;
        }
}
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].add){
            if (T[x].son[0]) Add(T[x].son[0],T[x].add);
            if (T[x].son[1]) Add(T[x].son[1],T[x].add);
            T[x].add=0;
        }
        if (T[x].lazy!=-INF){
            if (T[x].son[0]) Change(T[x].son[0],T[x].lazy);
            if (T[x].son[1]) Change(T[x].son[1],T[x].lazy);
            T[x].lazy=-INF;
        }
}
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;
        if (root[y]) {root[x]=true;root[y]=false;}
        else T[z].son[T[z].son[1]==y]=x;
        fa[x]=z;
        pushup(y);
}
void Prechange(int x){
        if (!root[x]) Prechange(fa[x]);
        pushdown(x);
}
void splay(int x){//to root
        Prechange(x);
        while (!root[x]){
            int y=fa[x],z=fa[y];
            int rx=T[y].son[0]==x,ry=T[z].son[0]==y;
            if (root[y]) rotate(x,rx);
            else{
```

```
                if (rx==ry) rotate(y,ry);
                else rotate(x,rx);
                rotate(x,ry);
            }
        }
        pushup(x);
}
int access(int x){//只有这条链上的是 mark 的
        int y=0;
        for (;x;x=fa[x]){
            splay(x);
            root[T[x].son[1]]=true;
            T[x].son[1]=y;
            root[y]=false;
            y=x;
            pushup(x);
        }
        return y;
}
bool judge(int u,int v){
        while (fa[u]) u=fa[u];
        while (fa[v]) v=fa[v];
        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);
        fa[u]=v;
        return 0;
}
bool cut(int u,int v){
        makeroot(u);
        splay(v);
        fa[T[v].son[0]]=fa[v];
        fa[v]=0;
        root[T[v].son[0]]=true;
        T[v].son[0]=0;
        pushup(v);
        return 0;
```

```
    }
    bool add(int u,int v,int val){
        makeroot(u);
        access(v);
        splay(v);
        Add(v,val);
        return 0;
    }
    bool change(int u,int v,int val){
        makeroot(u);
        access(v);
        splay(v);
        Change(v,val);
        return 0;
    }
    pair<int,int> ask(int u,int v){
        makeroot(u);
        access(v);
        splay(v);
        return make_pair(T[v].ans,T[v].cnt2);
    }
}T;
vector<int> edge[maxn];
void dfs(int x,int fa){
    T.fa[x]=fa;
    for (int v:edge[x]) if (v!=fa) dfs(v,x);
}
int n,m,TT;
int i,j,k;
int u,v;
int main(){
    int x=0;
    scanf("%d",&TT);
    while (TT--) {
        scanf("%d%d",&n,&m);
        FOR(i,1,n){
            int val;
            scanf("%d",&val);
            T.T[i].init(val);
        }
        FOR(i,1,n) T.root[i]=1;
        REP(i,n-1){
            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--){
            scanf("%d",&k);
            int x,y;
            if (k==1){
                int x0,y0;
                scanf("%d%d%d%d",&x,&y,&x0,&y0);
                T.cut(x,y);
                T.link(x0,y0);
            }else if (k==2){
                int val;
                scanf("%d%d%d",&x,&y,&val);
                T.change(x,y,val);
            }else if (k==3){
                int val;
                scanf("%d%d%d",&x,&y,&val);
                T.add(x,y,val);
            }else if (k==4){
                scanf("%d%d",&x,&y);
                pair<int,int> t=T.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();
    }
}
```

莫队

```
struct node{int l,r,id;}Q[maxn];//new direction
int pos[maxn];
LL ans[maxn],flag[maxn];
int a[maxn];
bool cmp(node a,node b){
    if (pos[a.l]==pos[b.l]) return a.r<b.r;
    return pos[a.l]<pos[b.l];
}
int n,m,k;   int i,j;
LL Ans;
int L=1,R=0;
```

```
void add(int x){
	Ans+=flag[a[x]^k];
	flag[a[x]]++; }
void del(int x){
	flag[a[x]]--;
	Ans-=flag[a[x]^k]; }
int main(){
	scanf("%d%d%d",&n,&m,&k);
	int sz=sqrt(n);
	FOR(i,1,n){
		scanf("%d",&a[i]);
		a[i]^=a[i-1];
		pos[i]=i/sz;
	}
	FOR(i,1,m){
		scanf("%d%d",&Q[i].l,&Q[i].r);
		Q[i].id=i;
	}
	sort(Q+1,Q+1+m,cmp);
	flag[0]=1;
	FOR(i,1,m){
		while (L<Q[i].l){del(L-1);L++;}
		while (L>Q[i].l){L--;add(L-1);}
		while (R<Q[i].r){R++;add(R);}
		while (R>Q[i].r){del(R);R--;}
		ans[Q[i].id]=Ans;
	}
	FOR(i,1,m) printf("%I64d\n",ans[i]);
}
```

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

```
int n,m,i,j,k,t;
vector<int>edge[N];
int used[N];
int matching[N];
/*注意数组的标号，必须满足二分图的条件
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 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;
}*/
int main(){
    int T;
    scanf("%d",&T);
    while (T--){
        scanf("%d%d",&n,&m);
        FOR(i,1,n){
            scanf("%d",&k);
            edge[i].clear();
            REP(j,k) scanf("%d",&t),edge[i].push_back(t+n);
```

```
            }
            if (BFS()==n) puts("YES");
            else puts("NO");
        }
}
```

最短路

Dijkstra（n^2）：

```
LL n,m,x;
LL a[N+2][N+2];
LL b[N+2];
bool vis[N+2];
LL i,j,k;
LL A,B,T;
int main()
{
    scanf("%lld%lld%lld",&n,&m,&x);
    FOR(i,n)
        FOR(j,n) a[i][j]=INF;
    FOR(i,m){
        scanf("%lld%lld%lld",&A,&B,&T);
        a[A][B]=T;
    }
    FOR(i,n) {b[i]=INF;vis[i]=0;}
    b[0]=INF;
    b[x]=0;
    int pos;
    FOR(i,n){
        pos=0;
        FOR(j,n) if (!vis[j]&&b[j]<b[pos]) pos=j;
        vis[pos]=1;
        FOR(j,n) if (!vis[j]&&b[pos]+a[pos][j]<b[j]) b[j]=b[pos]+a[pos][j];
    }
    FOR(i,n) printf("%lld ",b[i]);
}
```

Dijkstra（堆优化）：

```
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;//注意！！！
```

```
    }
};
vector<node> edge[maxn];//注意这里 priority_queue 是大根堆
int dis[maxn],n,m;
void dij(int s){//DIJKSTRA+HEAP
    int i;
    FOR(i,1,n) dis[i]=INF;
    dis[s]=0;
    priority_queue<node> Q;
    Q.push(node(s,dis[s]));
    while (!Q.empty()){
        node x=Q.top();Q.pop();
        REP(i,edge[x.n].size()){
            node y=edge[x.n][i];
            if (dis[y.n]>x.d+y.d){
                dis[y.n]=x.d+y.d;
                Q.push(node(y.n,dis[y.n]));
            }
        }
    }
}
```

SPFA BFS

```
vector<node> edge[maxn];
int dis[maxn],n,m;
bool vis[maxn];
int sumnum[maxn];//judge negative ring
bool spfa(int s){
    int i;
    FOR(i,1,n) dis[i]=INF;
    FOR(i,1,n) vis[i]=0;
    FOR(i,1,n) sumnum[i]=0;//judge negative ring
    dis[s]=0;
    deque<int> Q;//slf need
    Q.push_back(s);
//  int sum=0;//lll
    while (!Q.empty()){
        int u=Q.front();Q.pop_front();
//      if (!Q.empty()&&sum/Q.size()<dis[u]) Q.push_back(u);//lll
//      else {vis[u]=0; sum-=dis[u];}//lll
        vis[u]=0;//not lll
        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]){
                        vis[v.n]=1;
                        if (Q.empty()||dis[Q.front()]<dis[v.n]) Q.push_back(v.n);//slf
                        else Q.push_front(v.n);//slf
                        Q.push_back(v.n);//not slf
//                      sumnum[v.n]++;//judge negative ring
//                      if (sumnum[v.n]>=n) return 1;//judge negative ring
//                      sum+=dis[v.n];//lll
                    }
                }
            }
        }
//    return 0;//judge negative ring
}
```

SPFA DFS(只用于判负环)

```
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));
        }
        dij(1);
        FOR(i,2,n) printf("%d ",dis[i]==INF?-1:dis[i]);
        puts("");
    }
    return 0;
}
```

最小生成树

//Kruskal

```
struct node{
    int u,v,len;
    bool operator<(const node &A)const{
        if (len!=A.len) return len<A.len;
        if (u!=A.u) return u<A.u;
        return v<A.v;
    }
}Edge[maxn];
priority_queue<node> Q;
int fa[maxn];
inline void getfather(int x){
    if (x==fa[x]) return x;
    return fa[x]=getfather(fa[x]);
}
int n,m;
int main()
{
    scanf("%d%d",&n,&m);
    REP(i,m) scanf("%d%d%d",Edge[i].u,Edge[i].v,Edge[i].len);
    sort(Edge,Edge+m);
    while(Q.size()){
        edge=Edge[]();
        if (getfather(edge.u)==getfather(edge.v)) continue;
        fa[getfather[u]]=v;
        edge[u].push_back(v);
    }
}
```

强连通分量

```
vector<int> E[maxn];
int dfn[maxn],low[maxn],tot,n,ans=INF,cnt;
```

```
bool vis[maxn];
stack<int> S;
vector<int> V[maxn];
//u 割点:lowlink[u]>=dfn[v];
//uv 割边:lowlink[u]>dfn[v];
//块:lowlink[u]==dfn[v];
void tarjin(int x){
    low[x]=dfn[x]=++tot;
    S.push(x);vis[i]=1;
    for (int i=0;i<E[x].size();i++){
        int v=E[x][i];
        if (!dfn[v]){
            tarjin(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();S.pop();
            vis[now]=0;
            V[cnt].push_back(now);//改成 id[]=即可
            if (now==x) break;
        }
    }
}
```

网络流

//DINIC+当前弧优化

```
struct node{
    int to,cap,next;
    node(int t=0,int c=0,int n=0):to(t),cap(c),next(n){}
}edge[maxn*50];
int head[maxn];
int tot;
void addedge(int from,int to,int cap){
    edge[tot].to=to;
    edge[tot].next=head[from];
    edge[tot].cap=cap;
    head[from]=tot++;
    edge[tot].to=from;
```

```
    edge[tot].next=head[to];
    edge[tot].cap=0;
    head[to]=tot++;
}
queue<int> Q;
bool vis[maxn];
int d[maxn];
int cur[maxn];//当前弧优化
bool bfs(int s,int t){
    memset(vis,0,sizeof(vis));
    while (Q.size()) Q.pop();
    Q.push(s);
    d[s]=0;vis[s]=1;
    int i;
    while (!Q.empty()){
        int x=Q.front();Q.pop();
        for(i=head[x];i!=-1;i=edge[i].next){
            if (!vis[edge[i].to]&&edge[i].cap){
                vis[edge[i].to]=1;
                d[edge[i].to]=d[x]+1;
                Q.push(edge[i].to);
            }
        }
    }
    return vis[t];
}
int dfs(int x,int t,int flow){
    if (x==t||flow==0) return flow;
    int i,ret=0,f;
    for (i=cur[x];i!=-1;i=edge[i].next){
        if (d[x]+1==d[edge[i].to]&&((f=dfs(edge[i].to,t,min(flow,edge[i].cap)))>0)){
            edge[i].cap-=f;
            edge[i^1].cap+=f;
            ret+=f;
            flow-=f;
            cur[x]=i;
            if (flow==0) break;
        }
    }
    return ret;
}
int n,m,i;
int u,v,len,ans;
int s,t;
```

```
int main(){
    while (~scanf("%d%d",&n,&m)){
        memset(head,-1,sizeof(head));
        ans=0;tot=0;
        s=n+1;t=n+2;
        FOR(i,1,n){
            int a,b;
            scanf("%d%d",&a,&b);
            addedge(s,i,a);
            addedge(i,t,b);
        }
        FOR(i,1,m){
            scanf("%d%d%d",&u,&v,&len);
            addedge(u,v,len);
            addedge(v,u,len);
        }
        while (bfs(s,t)){
            int f;
            memcpy(cur,head,sizeof(head));
            while (f=dfs(s,t,INF)) ans+=f;
        }
        printf("%d\n",ans);
    }
}
```

最小费用流

```
//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
//注意观察特殊性质
struct node{
    LL to,cap,cost,rev;
    node(int t=0,int c=0,int n=0,int r=0):to(t),cap(c),cost(n),rev(r){}
};
vector<node> edge[maxn];
void addedge(int from,int to,LL cap,LL cost){
    edge[from].push_back(node(to,cap,cost,edge[to].size()));
    edge[to].push_back(node(from,0,-cost,edge[from].size()-1));
}
int n,m,V;
LL dis[maxn];
bool mark[maxn];
int pre_v[maxn],pre_e[maxn];
deque<int> Q;
pair<LL,LL> mincostflow(int s,int t,LL f){
```

```
    LL ret=0,d;
    int i,v;
    while (f){
        memset(dis,0x3f,sizeof(dis));
        memset(mark,0,sizeof(mark));
        while (Q.size()) Q.pop_front();
        dis[s]=0;Q.push_back(s);
        while (Q.size()){
            v=Q.front();mark[v]=0;Q.pop_front();
            REP(i,edge[v].size()){
                node &e=edge[v][i];
                if (e.cap>0&&dis[e.to]>dis[v]+e.cost){
                    dis[e.to]=dis[v]+e.cost;
                    pre_v[e.to]=v;
                    pre_e[e.to]=i;
                    if (!mark[e.to]){
                        if (Q.empty()||dis[Q.front()]<dis[e.to]) Q.push_back(e.to);
                        else Q.push_front(e.to);
                        mark[e.to]=1;
                    }
                }
            }
        }
        if (dis[t]==INFF) break;
        d=f;
        for (v=t;v!=s;v=pre_v[v])
            d=min(d,edge[pre_v[v]][pre_e[v]].cap);
        f-=d;
        ret+=d*dis[t];
        for (v=t;v!=s;v=pre_v[v]){
            node &e=edge[pre_v[v]][pre_e[v]];
            e.cap-=d;
            edge[v][e.rev].cap+=d;
        }
        if (d==0) break;
    }
    return make_pair(INFF-f,ret);
}
int i,j,k;
int main(){
    scanf("%d%d",&n,&m);
    FOR(i,1,m){
        LL u,v,c,w;
        scanf("%lld%lld%lld%lld",&u,&v,&c,&w);
```

```
        addedge(u,v,c,w);
    }V=n;
    pair<LL,LL> ans=mincostflow(1,n,INFF);
    printf("%lld %lld",ans.first,ans.second);
}
```

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
//最小流:ss->tt 流,t->s 连边,ss->tt 流
int n,m,q;
int i,j,k;
int ss,tt;
struct node{
    int to,cap,next;
}edge[maxn*3];
int tot;
int head[307];
int addedge(int from,int to,int cap){
    edge[tot].to=to;
    edge[tot].next=head[from];
    edge[tot].cap=cap;
    head[from]=tot++;
    edge[tot].to=from;
    edge[tot].next=head[to];
    edge[tot].cap=0;
    head[to]=tot++;
    return tot-1;//反的边  cap=正的  flow
}
bool vis[307];
int d[307];
queue<int> Q;
bool bfs(int s,int t){
    memset(vis,0,sizeof(vis));
    while (Q.size()) Q.pop();
    Q.push(s);
    d[s]=0;vis[s]=1;
    int i;
    while (Q.size()){
        int x=Q.front();Q.pop();
        for (i=head[x];i!=-1;i=edge[i].next){
            if (!vis[edge[i].to]&&edge[i].cap){
                vis[edge[i].to]=1;
```

```
                d[edge[i].to]=d[x]+1;
                Q.push(edge[i].to);
            }
        }
    }
    return vis[t];
}
int cur[307];//当前弧优化
int dfs(int x,int t,int flow){//dinic
    if (x==t||flow==0) return flow;
    int i,ret=0,f;
    for (i=cur[x];i!=-1;i=edge[i].next){
        if (d[x]+1==d[edge[i].to]&&(f=dfs(edge[i].to,t,min(flow,edge[i].cap)))>0){
            edge[i].cap-=f;
            edge[i^1].cap+=f;
            ret+=f;
            flow-=f;
            cur[x]=i;
            if (flow==0) break;
        }
    }
    return ret;
}
int in[307],out[307];
int add(int u,int v,int low,int high){
    int ret=addedge(u,v,high-low);
    out[u]+=low;in[v]+=low;
    return ret;
}
int sum,flow,E[maxn],ans[maxn];//E 为对应的边位置
int solve(){
    memset(head,0xff,sizeof(head));
    memset(in,0,sizeof(in));
    memset(out,0,sizeof(out));
    scanf("%d%d",&n,&m);
    flow=0;sum=0;tot=0;
    FOR(i,1,m){
        int u,v,low,high;
        scanf("%d%d%d%d",&u,&v,&low,&high);
        ans[i]=low;
        E[i]=add(u,v,low,high);//E[i]很有用
    }
    ss=n+1;tt=n+2;
    FOR(i,1,n){
```

```
            sum+=max(in[i]-out[i],0);
            if (in[i]>out[i]) addedge(ss,i,in[i]-out[i]);
            if (in[i]<out[i]) addedge(i,tt,out[i]-in[i]);
        }
        while (bfs(ss,tt)){
            int f;
            memcpy(cur,head,sizeof(head));
            while (f=dfs(ss,tt,INF)) flow+=f;
        }
        if (flow!=sum) return 0*puts("NO");
        else {
            puts("YES");
            FOR(i,1,m){
                ans[i]+=edge[E[i]].cap;
                printf("%d\n",ans[i]);
            }
        }
}
int main()
{
        int T;
        scanf("%d",&T);
        while (T--){
            solve();
        }
}
```

上下界费用流

// Hihocoder 1424，限制很多的一道题，只是留板子

```
struct node{
        LL to,cap,cost,rev;
        node(int t=0,int c=0,int n=0,int r=0):to(t),cap(c),cost(n),rev(r){}
};
vector<node> edge[maxn];
void addedge(int from,int to,LL cap,LL cost){
        edge[from].push_back(node(to,cap,cost,edge[to].size()));
        edge[to].push_back(node(from,0,-cost,edge[from].size()-1));
}
LL dis[maxn];
bool mark[maxn];
int pre_v[maxn],pre_e[maxn];
deque<int> Q;
pair<int,int> mincostflow(int s,int t,int f){
```

```
    int ret=0,d;
    int i,v;
    while (f){
        memset(dis,0x3f,sizeof(dis));
        memset(mark,0,sizeof(mark));
        while (Q.size()) Q.pop_front();
        dis[s]=0;Q.push_back(s);
        while (Q.size()){
            v=Q.front();mark[v]=0;Q.pop_front();
            REP(i,edge[v].size()){
                node &e=edge[v][i];
                if (e.cap>0&&dis[e.to]>dis[v]+e.cost){
                    dis[e.to]=dis[v]+e.cost;
                    pre_v[e.to]=v;
                    pre_e[e.to]=i;
                    if (!mark[e.to]){
                        if (Q.empty()||dis[Q.front()]<dis[e.to]) Q.push_back(e.to);
                        else Q.push_front(e.to);
                        mark[e.to]=1;
                    }
                }
            }
        }
        if (dis[t]==INF) break;
        d=f;
        for (v=t;v!=s;v=pre_v[v])
            d=min(d,edge[pre_v[v]][pre_e[v]].cap);
        f-=d;
        ret+=d*dis[t];
        for (v=t;v!=s;v=pre_v[v]){
            node &e=edge[pre_v[v]][pre_e[v]];
            e.cap-=d;
            edge[v][e.rev].cap+=d;
        }
        if (d==0) break;
    }
    return make_pair(INF-f,ret);
}

int n,m;
int i,j;
int VAL[57][57];
int addrow[57][57];
int addcol[57][57];
```

```
int row[57],col[57];
int in[maxn],out[maxn];
int u,v;
int s,t,S,T;
int tot;
int sum;
void add(int u,int v,int low,int high,int cost){
    addedge(u,v,high-low,cost);
    out[u]+=low;in[v]+=low;
}
void solve(int n){
    tot=0;
    FOR(i,1,n) row[i]=++tot;
    FOR(i,1,n) col[i]=++tot;
    s=++tot;t=++tot;
    S=++tot;T=++tot;
    FOR(i,1,n)
        FOR(j,1,n) scanf("%d",&VAL[i][j]);
    FOR(i,1,n){
        int cnt=0;
        FOR(j,1,n) cnt+=VAL[i][j];
        add(s,row[i],cnt,cnt,0);
        cnt=0;
        FOR(j,1,n) cnt+=VAL[j][i];
        add(s,col[i],cnt,cnt,0);
    }
    FOR(i,1,n){
        int l,r;
        scanf("%d%d",&l,&r);
        add(row[i],t,l,r,0);
    }
    FOR(i,1,n){
        int l,r;
        scanf("%d%d",&l,&r);
        add(col[i],t,l,r,0);
    }
    FOR(i,1,n)
        FOR(j,1,n) addrow[i][j]=addcol[i][j]=0;
    REP(i,n*n/2){
        int x0,y0,x1,y1;
        scanf("%d%d%d%d",&x0,&y0,&x1,&y1);
        if (VAL[x0][y0]==VAL[x1][y1]) continue;
        if (VAL[x0][y0]==1){
            if (y0==y1) addrow[x0][x1]++;
```

```
                else addcol[y0][y1]++;
            }else if (VAL[x1][y1]==1){
                if (y0==y1) addrow[x1][x0]++;
                else addcol[y1][y0]++;
            }
        }
        FOR(i,1,n){
            FOR(j,1,n){
                if (addrow[i][j]) add(row[i],row[j],0,addrow[i][j],1);
                if (addcol[i][j]) add(col[i],col[j],0,addcol[i][j],1);
            }
        }
        sum=0;
        add(t,s,0,INF,0);
        FOR(i,1,tot){
            sum+=max(in[i]-out[i],0);
            if (in[i]>out[i]) addedge(S,i,in[i]-out[i],0);
            if (in[i]<out[i]) addedge(i,T,out[i]-in[i],0);
        }
        pair<int,int> now=mincostflow(S,T,INF);
        if (now.first!=sum) puts("-1");
        else printf("%d\n",now.second);
        FOR(i,1,tot) edge[i].clear();
        FOR(i,1,tot) in[i]=out[i]=0;
}
int main()
{
        while (~scanf("%d",&n)) solve(n);
}
```

树分治

//乘积立方数个数，如果是 sum 直接枚举其实就好

```
LL K;
LL MUL[37];
LL getSum(LL x,LL y){
        LL ret=0,i;
        REP(i,K) ret=ret+(x/MUL[i]%3+y/MUL[i]%3)%3*MUL[i];
        return ret;
}
LL getDiv(LL x){
        LL ret=0,i;
        REP(i,K) ret=ret+(3-x/MUL[i]%3)%3*MUL[i];
        return ret;
```

```
}
LL color[maxn];
vector<int> edge[maxn];
LL ans;
int size[maxn];
bool mark[maxn];
int minweight,root;
void dfs1(int x,int fa,int n){
	int weight=0;
	size[x]=1;
	for (int v:edge[x]){
		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;}
}
map<LL,int> now;
map<LL,int> MP;
void dfs2(int x,int fa,LL num){
	now[getSum(color[x],num)]++;
	for (int v:edge[x]){
		if (v==fa||mark[v]) continue;
		dfs2(v,x,getSum(num,color[x]));
	}
}
void calc(int x){
	MP.clear();
	MP[color[x]]++;
	for (int u:edge[x]){
		if (mark[u]) continue;
		now.clear();
		dfs2(u,0,0);
		for(pair<LL,int> P:now) ans+=MP[getDiv(P.first)]*P.second;
		for(pair<LL,int> P:now) MP[getSum(color[x],P.first)]+=P.second;
	}
	MP.clear();
}
void dfs3(int x){
	mark[x]=1;
	calc(x);
	for (int v:edge[x]){
```

```
            if (mark[v]) continue;
            minweight=size[v];
            dfs1(v,0,size[v]);
            dfs3(root);
        }
}
int n,m;
LL C[maxn];
LL P;
int main(){
        int i,j;
        MUL[0]=1;
        FOR(i,1,33) MUL[i]=MUL[i-1]*3;
        while (~scanf("%d",&n)){
            ans=0;
            scanf("%d",&K);
            REP(i,K) scanf("%lld",&C[i]);
            FOR(i,1,n){
                scanf("%lld",&P);
                REP(j,K){
                    int t=0;
                    while (P%C[j]==0){
                        P/=C[j];
                        t++;
                        if (t==3) t=0;
                    }
                    color[i]+=MUL[j]*t;
                }
                if (color[i]==0) ans++;
            }
            REP(i,n-1){
                int u,v;
                scanf("%d%d",&u,&v);
                edge[u].push_back(v);
                edge[v].push_back(u);
            }
            minweight=n;
            dfs1(1,0,n);
            dfs3(root);
            printf("%lld\n",ans);
            FOR(i,1,n) mark[i]=0;
            FOR(i,1,n) color[i]=0;
            FOR(i,1,n) vector<int>().swap(edge[i]);
        }
```

```
}
```

部分树上 dp

到叶结点最大距离

```
void dfs1(int u,int from){
    int v,w,i;
    REP(i,edge[u].size()){
        v=edge[u][i].first;
        if (v==from) continue;
        w=edge[u][i].second;
        dfs1(v,u);
        if (l1[u]<l1[v]+w) l2[u]=l1[u],l1[u]=l1[v]+w,son[u]=v;
        else if (l2[u]<l1[v]+w) l2[u]=l1[v]+w;
    }
}
void dfs2(int u,int from,LL d){//从叶子开始
    int v,w,i;
    len[u]=max(d,l1[u]);
    REP(i,edge[u].size()){
        v=edge[u][i].first;
        if (v==from) continue;
        w=edge[u][i].second;
        if (son[u]==v) dfs2(v,u,max(d,l2[u])+w);
        else dfs2(v,u,max(d,l1[u])+w);
    }
}
```

另一种方法

```
void dfs1(int u,int x,int length){//需要好多次(findmaxlen)
    int i;
    if (length>len[u]) len[u]=length;
    if (length>mxlen) mx=u,mxlen=length;
    REP(i,edge[u].size())
        if (edge[u][i]!=x) dfs1(edge[u][i],u,length+1);
}
void dfs2(int x,int father){
    int i;
    root[x]=father;
    value[father].push_back(len[x]);
    num[father]++;
    REP(i,edge[x].size())
        if (!root[edge[x][i]]) dfs2(edge[x][i],father);
}
```

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)

```
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;
}
```

树上距离除 k 向上取整

```
LL count[maxn][6];
vector<int> edge[maxn];
LL num[maxn],cnt[maxn];//端点,满足条件的次数
int k;
LL ans;
void dfs(int u,int from){
    int i,j,c1,c2;
    count[u][0]=1;
    cnt[u]=1;
    REP(i,edge[u].size()){
        int v=edge[u][i];
        if (from==v) continue;
        dfs(v,u);
        REP(c1,k)
            REP(c2,k){
                ans+=count[u][c1]*count[v][c2];
                if (c1+c2+1>k) ans+=count[u][c1]*count[v][c2];
            }
        ans+=cnt[u]*num[v]+num[u]*cnt[v];
        num[u]+=num[v]+count[v][k-1];
        cnt[u]+=cnt[v];
        REP(c1,k) count[u][c1]+=count[v][(c1-1+k)%k];
    }
}
```

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 Tsat{
    vector<int> edge[maxn*2];
    stack<int> S;
```

```
    int belong[maxn*2];
    int dfn[maxn*2],low[maxn*2];
    bool vis[maxn*2];
    int tot,cnt;
    bool mark;
    void init(int n){
        tot=cnt=0;
        int i;
        REP(i,n*2) edge[i].clear();
        REP(i,n*2) dfn[i]=vis[i]=low[i]=belong[i]=0;
    }
    void dfs(int u){
        int i;
        dfn[u]=low[u]=++tot;
        S.push(u);vis[u]=1;
        REP(i,edge[u].size()){
            int v=edge[u][i];
            if (!dfn[v]){
                dfs(v);
                low[u]=min(low[u],low[v]);
            }else if (vis[v]){
                low[u]=min(low[u],dfn[v]);
            }
        }
        if (dfn[u]==low[u]){
            cnt++;
            while (1){
                int now=S.top();S.pop();
                vis[now]=0;
                belong[now]=cnt;
                if (now==u) break;
            }
        }
    }
    inline void addedge(int u,int v){
        edge[u].push_back(v);
    }
    bool solve(int n){
        int i;
        REP(i,n*2) if (!dfn[i]) dfs(i);
        REP(i,n) if (belong[i]==belong[i+n]) return 0;
        return 1;
    }
}sat;
```

```
int n,m,t;
int numA,numB;
int A[maxn][2],B[maxn][2];
int i,j;
int tot;
struct node{
    int x,y;
}S1,S2,a[maxn];
inline int dist(node A,node B){
    return abs(A.x-B.x)+abs(A.y-B.y);
}
void preadd(){
    int i,u,v;
    REP(i,numA){
        u=A[i][0];v=A[i][1];
        sat.addedge(u,v+n);sat.addedge(u+n,v);
        sat.addedge(v,u+n);sat.addedge(v+n,u);
    }
    REP(i,numB){
        u=B[i][0];v=B[i][1];
        sat.addedge(u,v);sat.addedge(u+n,v+n);
        sat.addedge(v,u);sat.addedge(v+n,u+n);
    }
}
bool solve(int x){
    sat.init(n);
    preadd();
    int i,j;
    REP(i,n)
        rep(j,i+1,n){
        if (dist(a[i],S1)+dist(a[j],S1)>x) {sat.addedge(i,j+n);sat.addedge(j,i+n);}
        if (dist(a[i],S2)+dist(a[j],S2)>x) {sat.addedge(i+n,j);sat.addedge(j+n,i);}
        if (dist(a[i],S1)+dist(a[j],S2)+dist(S1,S2)>x) {sat.addedge(i,j);sat.addedge(j+n,i+n);}
        if (dist(a[i],S2)+dist(a[j],S1)+dist(S1,S2)>x) {sat.addedge(i+n,j+n);sat.addedge(j,i);}
    }
    return sat.solve(n);
}
int l,r,mid;
int main(){
    int t,m;
    while (~scanf("%d%d%d",&n,&numA,&numB)){
        scanf("%d%d%d%d",&S1.x,&S1.y,&S2.x,&S2.y);
```

```
        REP(i,n) scanf("%d%d",&a[i].x,&a[i].y);
        REP(i,numA) {scanf("%d%d",&A[i][0],&A[i][1]);A[i][0]--;A[i][1]--;}/*careful!!!*/
        REP(i,numB) {scanf("%d%d",&B[i][0],&B[i][1]);B[i][0]--;B[i][1]--;}/*careful!!!*/
        l=-1;r=5000000;
        while (l+1<r){
            mid=(r+l)/2;
            if (!solve(mid)) l=mid;
            else r=mid;
        }
        if (l<4500000) printf("%d\n",l+1);
        else printf("-1\n");
    }
}
```

dfs 序

//常用方法：时间戳、莫队、拆开操作

```
void dfs(int u,int from){
    int v,i;
    in[u]=++tot;
    REP(i,edge[u].size()){
        v=edge[u][i];
        if (v==from) continue;
        dfs(v,u);
    }
    out[u]=tot;
}
```

树链剖分

难题(区间合并)

```
int tot;
struct node{
    int lval,rval,ldown,lup,rdown,rup,upmx,downmx;
    node():upmx(0),downmx(0){};
}tree[maxn<<2];
int a[maxn];
node merge(node L,node R){
    if (L.upmx==0) return R;
    if (R.upmx==0) return L;
    node ret;
    ret.upmx=max(L.upmx,R.upmx);
    ret.downmx=max(L.downmx,R.downmx);
    ret.lval=L.lval;
```

```
    ret.lup=L.lup;
    ret.ldown=L.ldown;
    ret.rval=R.rval;
    ret.rup=R.rup;
    ret.rdown=R.rdown;
    if (L.rval<R.lval){
        ret.upmx=max(ret.upmx,L.rup+R.lup);
        if (L.downmx==1) ret.lup=L.lup+R.lup;
        if (R.downmx==1) ret.rup=L.rup+R.rup;
    }
    if (L.rval>R.lval){
        ret.downmx=max(ret.downmx,L.rdown+R.ldown);
        if (L.upmx==1) ret.ldown=L.ldown+R.ldown;
        if (R.upmx==1) ret.rdown=L.rdown+R.rdown;
    }
    return ret;
}
void build(int x,int l,int r){
    if (l==r){
        tree[x].lval=tree[x].rval=a[l];

tree[x].lup=tree[x].ldown=tree[x].rup=tree[x].rdown=tree[x].upmx=tree[x].downmx=1;
        return;
    }
    int mid=(l+r)/2;
    build(x<<1,l,mid);
    build(x<<1|1,mid+1,r);
    tree[x]=merge(tree[x<<1],tree[x<<1|1]);
}
node query(int x,int l,int r,int L,int R){
    node ret;
    if (l<=L&&R<=r) return tree[x];
    int mid=(L+R)/2;
    if (mid>=l&&r>mid) return merge(query(x<<1,l,r,L,mid),query(x<<1|1,l,r,mid+1,R));
    if (mid>=l) return query(x<<1,l,r,L,mid);
    return query(x<<1|1,l,r,mid+1,R);
}
int n,i,j,q;
int u,v;
vector<int> edge[maxn];
int fa[maxn],son[maxn],top[maxn],dep[maxn],id[maxn],sz[maxn];
int b[maxn];
void dfs1(int u,int depth){
    int v,i,mx=-1;
```

```
    son[u]=0;sz[u]=1;dep[u]=depth;
    REP(i,edge[u].size()){
        v=edge[u][i];
        dfs1(v,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){//这里需要注意方向
    node up,down;
    int ret,mark1=0,mark2=0;
    while (top[x]!=top[y]){
        if (dep[top[x]]>dep[top[y]]){
            up=merge(query(1,id[top[x]],id[x],1,tot),up);
            x=fa[top[x]];
            mark1=1;
        }else {
            down=merge(query(1,id[top[y]],id[y],1,tot),down);
            y=fa[top[y]];
            mark2=1;
        }
    }
    if (dep[x]>dep[y]) up=merge(query(1,id[y],id[x],1,tot),up),mark1=1;
    else down=merge(query(1,id[x],id[y],1,tot),down),mark2=1;
    ret=max(up.downmx,down.upmx);
    if (mark1&&mark2&&up.lval<down.lval) ret=max(ret,up.ldown+down.lup);
    return ret;
}
int T,t;
int main(){
    scanf("%d",&T);
    FOR (t,1,T){
        scanf("%d",&n);
        FOR(i,1,n) edge[i].clear();tot=0;
```

```
        FOR(i,1,n) scanf("%d",&b[i]);
        FOR(i,2,n){scanf("%d",&fa[i]); edge[fa[i]].push_back(i);}
        dfs1(1,1);
        dfs2(1,1);
        FOR(i,1,n) a[id[i]]=b[i];
        build(1,1,tot);
        scanf("%d",&q);
        printf("Case #%d:\n",t);
        while (q--){
            scanf("%d%d",&u,&v);
            printf("%d\n",Query(u,v));
        }
        if (t!=T) puts("");
    }
}
```

树链剖分求 LCA

```
vector<int> edge[maxn];
int sz[maxn],fa[maxn],son[maxn],top[maxn],dep[maxn],id[maxn];//id 没用
int tot=0;
void dfs1(int u,int depth){
    int v,i,mx=-1;
    sz[u]=1;dep[u]=depth;son[u]=0;
    for(int v:edge[u]){
        dfs1(v,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);
    for (int v:edge[u]){
        if (v==son[u]) continue;
        dfs2(v,v);
    }
}
int query(int x,int y){
    while (top[x]!=top[y]){
        if (dep[top[x]]<dep[top[y]]) swap(x,y);
        x=fa[top[x]];
    }
```

```
    if (dep[x]>dep[y]) swap(x,y);
    return x;
}
int len(int x,int y){
    return dep[x]+dep[y]-dep[query(x,y)]*2+1;//point
}
```

离线 tarjin 求 LCA

```
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;
int i,k;
int 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]);
}
int T;
int main(){
    scanf("%d",&T);
    while (T--) solve();
}
```

数学相关

```
void getPrim(){//线性的筛法求素数
	int o=0;
	register int i,j;
	FOR(i,2,Nmax){
		if (!prim[i]) prim[++prim[0]]=i;
		FOR(j,1,prim[0]){
			if (i*prim[j]>Nmax) break;
			prim[i*prim[j]]=1;
			if (i%prim[j]==0) break;
		}
	}
}
```

逆元

```
int n,m;
int i,j,k;
//d==1 时存在逆元 //(x+p)%p 为逆元//d!=1 可用 num*a/d 来代替逆元(num|d)
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 getinv(int n){
	if (n==1) return 1;
	return (M-M/n)*(getinv(M%n))%M;
}
LL inv1[1000002];
LL inv2[1000002];
LL inv3[1000002];
int main()
{
	LL d,x,y;
//	FOR(i,1,1000000) {exgcd(i,M,d,inv[i],y); inv1[i]=(inv[i]+M)%M;}
//	FOR(i,1,1000000) inv2[i]=getinv(i);
	inv3[0]=inv3[1]=1;
	FOR(i,2,1000000) inv3[i]=(M-M/i)*inv3[M%i]%M;
//	FOR(i,1,1000000) printf("%lld	",inv3[i]*i%M);
}
```

C(n,n)

```
int n,m;
int i,j,k;
```

```
LL inv[1000002];//inverse
LL fac[1000002];//Factorial
void init(){
	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;
}
LL C(int n,int m){
	return fac[n]*inv[m]%M*inv[n-m]%M; }
int main()
{
	LL d,x,y;
	init();
	printf("%d",C(10,3));
}
```

Lucas Cnn

```
int n,m;
int i,j,k;
LL inv[1000002];//inverse
LL fac[1000002];//Factorial
void init(){
	int i;
	fac[0]=1;
	FOR(i,1,1000000) fac[i]=i*fac[i-1]%MOD;
	inv[0]=inv[1]=1;
	FOR(i,2,1000000) inv[i]=(MOD-MOD/i)*inv[MOD%i]%MOD;
	FOR(i,1,1000000) inv[i]=inv[i]*inv[i-1]%MOD;
}
LL C(int n,int m){
	return fac[n]*inv[m]%MOD*inv[n-m]%MOD;
}
LL lucas(LL n,LL m){//注意 MOD 不能太大=_=!
	return m==0?1:1ll*C(n%MOD,m%MOD)*lucas(n/MOD,m/MOD)%MOD;
}
int main()
{
	LL d,x,y;
	init();
	printf("%d",lucas(10,3));
}
```

数位 dp

对于某一个问题，f[i][j][k][l]表示 i 位,第一位 j,k=0/1(表示是否满足条件),余数或者其他为 l 时的情况个数

```
LL n,m;
LL dp[20][3];//0:
LL i,j,k;
void init(){
    memset(dp,0,sizeof(dp));
    dp[0][0]=1;
    FOR(i,1,10){
        dp[i][0]=dp[i-1][0]*9-dp[i-1][1];//okay
        dp[i][1]=dp[i-1][0];//2.....
        dp[i][2]=dp[i-1][0]+dp[i-1][1]+dp[i-1][2]*10;//not okay
    }
}
int A[20];
int calc(int a){
    int sum=a;
    int m=0;
    int ans=0;
    bool flag=false;
    while(a){
        A[++m]=a%10;
        a/=10;
    }
    A[m+1]=0;
    for (int i=m;i>=1;i--){
        ans+=dp[i-1][2]*A[i];
        if (flag){
            ans+=dp[i-1][0]*A[i];
        }else{
            if (A[i]>4) ans+=dp[i-1][0];
            if (A[i+1]==6&&A[i]>2) ans+=dp[i][1];
            if (A[i]>6) ans+=dp[i-1][1];
            if (A[i]==4||A[i]==2&&A[i+1]==6) flag=1;
        }
    }
    if (flag) ans++;
    return sum-ans;
}
int main(){
    int a,b;
    int l,r;
```

```
    init();
    while (~scanf("%d%d",&l,&r)&&(l||r)) printf("%d\n",calc(r)-calc(l-1));
}
```

博弈：NIM,SG

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){
//  printf("%d %d\n",x,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;
}*/
int main(){
    sg[0]=0;
}
```

求凸包

```
struct node{
	double x,y;
	bool operator <(const node &a) const{
		if (y<a.y) return 1; if (y>a.y) return 0;
		return x<a.x;
	}
}p[maxn],P[maxn];
inline double X(node A,node B,node C){ return (B.x-A.x)*(C.y-A.y)-(B.y-A.y)*(C.x-A.x); }
inline double len(node A,node B){ return sqrt((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y)); }
bool cmp(node A,node B){
	double cp=X(p[0],A,B);
	if (cp>0) return 1;if (cp<0) return 0;
	return len(p[0],A)<len(p[0],B);
}
int n,m;
double t;
int tot;
int i,j,k;
double ans;//求长度的
int main(){
	while (~scanf("%d%lf",&n,&t)){
		REP(i,n) scanf("%lf%lf",&p[i].x,&p[i].y);
//		  ans=2*pi*t;//没啥用//=0
		if (n==1) printf("%.0lf",ans);
		else if (n==2) printf("%.0lf",ans+len(p[0],p[1]));
		else {
			REP(i,n) if (p[i]<p[0]) swap(p[0],p[i]);
			sort(p+1,p+n,cmp);
			P[0]=p[0];
			P[1]=p[1];
			tot=1;
			rep(i,2,n){
				while (tot>0&&X(P[tot-1],P[tot],p[i])<=0) tot--;
				P[++tot]=p[i];
			}
			REP(i,tot) ans+=len(P[i],P[i+1]);
			ans+=len(P[0],P[tot]);
			printf("%.0lf",ans);
		}puts("");
	}
}
```