```cpp
//输入外挂
//扩展欧几里得
//素数线性筛
//中国剩余定理
//逆元
//FFT
//NTT
//MILLER_RABIN
//AC自动机
//数位DP
//点分治
//线性递推模板m^2logn
//吉司机线段树
//树链剖分
//二分图匹配
//决策单调性
//bitset
//树状数组区间修改区间查询
//SA

//输入外挂
#include <bits/stdc++.h>
using namespace std;
template <class T>
inline bool scan_d(T &ret){
    char c; int sgn;
    if(c=getchar(), c==EOF)return 0;
    while(c!='-'&&(c<'0'||c>'9'))c=getchar();
    sgn=(c=='-')?-1:1;
    ret=(c=='-')?0:(c-'0');
    while(c=getchar(), c>='0'&&c<='9')ret=ret*10+c-'0';
    ret*=sgn;
    return 1;
}

int main(){
    int a, b;
    while(1){
        scan_d(a);
        scan_d(b);
        cout<<a+b<<endl;
    }
}


//扩展欧几里得
//求ax+by=gcd(a, b)的解
void exgcd(ll a, ll b, ll &x, ll &y){
    if(b==0){
        x=1, y=0;
        return ;
    }
    exgcd(b, a%b, x, y);
    ll tmp=x;
    x=y;
    y=tmp-(a/b)*y;
}

//素数线性筛
const int maxn=205000;

bool vis[maxn];
int prime[maxn];
int tot;
void init(){
    memset(vis, false, sizeof(vis));
    tot=0;
    for(int i=2;i<maxn;i++){
        if(!vis[i])prime[++tot]=i;
        for(int j=1;j<=tot&&prime[j]*i<maxn;j++){
            vis[prime[j]*i]=true;
            if(i%prime[j]==0)break;
        }
```

```
    }
}

//中国剩余定理
//正整数m1, m2, ..., mk两两互素, 则同余方程组
//
//x%m1==a1
//x%m2==a2
//
//x%mk==ak
//
//在模M=m1*m1*...*mk下的解唯一
//
//x%M==sigma(ai*Mi*inv(Mi))
//
//Mi=M/mi, inv(Mi)为Mi模mi的逆元
int CRT(int a[], int m[], int n){
    int M=1;
    int ans=0;
    for(int i=1;i<=n;i++)M*=m[i];
    for(int i=1;i<=n;i++){
        int x, y;
        int Mi=M/m[i];
        exgcd(Mi, m[i], x, y);//m[i]不一定是素数, 故要用exgcd求逆元
        ans=(ans+a[i]*Mi*x)%M;
    }
    if(ans<0)ans+=M;
    return ans;
}

//卢卡斯定理
//rm to do

//逆元
//ans=(a/b)%m=(a%(mb))/b
//fermat小定理
//exgcd
//o(n)预处理mod的逆元
int mod;
int inv[maxn];

void init(){
    inv[1] = 1;
    for(int i=2;i<mod;i++){
        inv[i]=inv[mod%i]*(mod-mod/i)%mod;
    }
}

//欧拉函数
//rm to do

//莫比乌斯函数
//rm to do

//威佐夫博弈
//rm to do

//反NIM博弈
//rm to do

//FFT
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <algorithm>
#include <math.h>
using namespace std;

const double PI = acos(-1.0);
//复数结构体
struct complex
{
    double r,i;
```

```
147         complex(double _r = 0.0,double _i = 0.0)
148         {
149             r = _r; i = _i;
150         }
151         complex operator +(const complex &b)
152         {
153             return complex(r+b.r,i+b.i);
154         }
155         complex operator -(const complex &b)
156         {
157             return complex(r-b.r,i-b.i);
158         }
159         complex operator *(const complex &b)
160         {
161             return complex(r*b.r-i*b.i,r*b.i+i*b.r);
162         }
163     };
164     /*
165      * 进行FFT和IFFT前的反转变换。
166      * 位置i和 （i二进制反转后位置）互换
167      * len必须去2的幂
168      */
169     void change(complex y[],int len)
170     {
171         int i,j,k;
172         for(i = 1, j = len/2;i < len-1; i++)
173         {
174             if(i < j)swap(y[i],y[j]);
175             //交换互为小标反转的元素，i<j保证交换一次
176             //i做正常的+1，j左反转类型的+1,始终保持i和j是反转的
177             k = len/2;
178             while( j >= k)
179             {
180                 j -= k;
181                 k /= 2;
182             }
183             if(j < k) j += k;
184         }
185     }
186     /*
187      * 做FFT
188      * len必须为2^k形式，
189      * on==1时是DFT，on==-1时是IDFT
190      */
191     void fft(complex y[],int len,int on)
192     {
193         change(y,len);
194         for(int h = 2; h <= len; h <<= 1)
195         {
196             complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
197             for(int j = 0;j < len;j+=h)
198             {
199                 complex w(1,0);
200                 for(int k = j;k < j+h/2;k++)
201                 {
202                     complex u = y[k];
203                     complex t = w*y[k+h/2];
204                     y[k] = u+t;
205                     y[k+h/2] = u-t;
206                     w = w*wn;
207                 }
208             }
209         }
210         if(on == -1)
211             for(int i = 0;i < len;i++)
212                 y[i].r /= len;
213     }
214     const int MAXN = 200010;
215     complex x1[MAXN],x2[MAXN];
216     char str1[MAXN/2],str2[MAXN/2];
217     int sum[MAXN];
218     int main()
219     {
```

```
    while(scanf("%s%s",str1,str2)==2)
    {
        int len1 = strlen(str1);
        int len2 = strlen(str2);
        int len = 1;
        while(len < len1*2 || len < len2*2)len<<=1;
        for(int i = 0;i < len1;i++)
            x1[i] = complex(str1[len1-1-i]-'0',0);
        for(int i = len1;i < len;i++)
            x1[i] = complex(0,0);
        for(int i = 0;i < len2;i++)
            x2[i] = complex(str2[len2-1-i]-'0',0);
        for(int i = len2;i < len;i++)
            x2[i] = complex(0,0);
        //求DFT
        fft(x1,len,1);
        fft(x2,len,1);
        for(int i = 0;i < len;i++)
            x1[i] = x1[i]*x2[i];
        fft(x1,len,-1);
        for(int i = 0;i < len;i++)
            sum[i] = (int)(x1[i].r+0.5);
        for(int i = 0;i < len;i++)
        {
            sum[i+1]+=sum[i]/10;
            sum[i]%=10;
        }
        len = len1+len2-1;
        while(sum[len] <= 0 && len > 0)len--;
        for(int i = len;i >= 0;i--)
            printf("%c",sum[i]+'0');
        printf("\n");
    }
    return 0;
}

//NTT
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
#define MAXN 262144

const long long P=50000000001507329LL; // 190734863287 * 2 ^ 18 + 1
//const int P=1004535809; // 479 * 2 ^ 21 + 1
//const int P=998244353; // 119 * 2 ^ 23 + 1
const int G=3;

long long mul(long long x,long long y)
{
    return (x*y-(long long)(x/(long double)P*y+1e-3)*P+P)%P;
}
long long qpow(long long x,long long k,long long p)
{
    long long ret=1;
    while(k)
    {
        if(k&1) ret=mul(ret,x);
        k>>=1;
        x=mul(x,x);
    }
    return ret;
}

long long wn[25];
void getwn()
{
    for(int i=1; i<=18; ++i)
    {
        int t=1<<i;
        wn[i]=qpow(G,(P-1)/t,P);
    }
}
```

```
293
294 int len;
295 void NTT(long long y[],int op)
296 {
297     for(int i=1,j=len>>1,k; i<len-1; ++i)
298     {
299         if(i<j) swap(y[i],y[j]);
300         k=len>>1;
301         while(j>=k)
302         {
303             j-=k;
304             k>>=1;
305         }
306         if(j<k) j+=k;
307     }
308     int id=0;
309     for(int h=2; h<=len; h<<=1)
310     {
311         ++id;
312         for(int i=0; i<len; i+=h)
313         {
314             long long w=1;
315             for(int j=i; j<i+(h>>1); ++j)
316             {
317                 long long u=y[j],t=mul(y[j+h/2],w);
318                 y[j]=u+t;
319                 if(y[j]>=P) y[j]-=P;
320                 y[j+h/2]=u-t+P;
321                 if(y[j+h/2]>=P) y[j+h/2]-=P;
322                 w=mul(w,wn[id]);
323             }
324         }
325     }
326     if(op==-1)
327     {
328         for(int i=1; i<len/2; ++i) swap(y[i],y[len-i]);
329         long long inv=qpow(len,P-2,P);
330         for(int i=0; i<len; ++i) y[i]=mul(y[i],inv);
331     }
332 }
333 void Convolution(long long A[],long long B[],int len1,int len2)
334 {
335     int n=max(len1,len2);
336     for(len=1; len<(n<<1); len<<=1);
337     for(int i=len1; i<len; ++i)
338     {
339         A[i]=0;
340     }
341     for (int i=len2;i<len;i++)
342     {
343         B[i]=0;
344     }
345
346     NTT(A,1);
347     NTT(B,1);
348     for(int i=0; i<len; ++i)
349     {
350         A[i]=mul(A[i],B[i]);
351     }
352     NTT(A,-1);
353 }
354
355 long long A[MAXN],B[MAXN];
356 char s1[MAXN],s2[MAXN];
357 void debug(){
358     A[0]=1, A[1]=2, A[2]=3;
359     B[0]=1, B[1]=2, B[2]=3;
360     Convolution(A, B, 3, 3);
361     for(int i=0;i<=6;i++)printf("%lld\n", A[i]);
362 }
363 int main()
364 {
365     getwn();
```

```
    debug();
}

//FWT
//rm to do

//Miller Rabin
//Pollard_rho

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

typedef unsigned long long ll;

const int T=10;

ll mul(ll x,ll y, ll P)
{
    return (x*y-(ll)(x/(ll)P*y+1e-3)*P+P)%P;
}

ll exp(ll x, ll y, ll mod){
    ll ans=1;
    ll base=x;
    while(y){
        if(y&1){
            ans=mul(ans, base, mod);
        }
        y>>=1;
        base=mul(base, base, mod);
    }
    return ans;
}

bool miller_rabin(ll n)
{
    if(n==2)return true;
    if(n<2||!(n&1))return false;
    ll m=n-1;
    ll k=0;
    while(!(m&1)){
        k++;
        m>>=1;
    }
    for(int i=0;i<T;i++){
        ll a=rand()%(n-1)+1;
        ll x=exp(a, m, n);
        ll y=0;
        for(int j=1;j<=k;j++){
            y=mul(x, x, n);
            if(y==1&&x!=1&&x!=n-1)return false;
            x=y;
        }
        if(y!=1)return false;
    }
    return true;
}

ll Pollard_rho(ll x,ll c){
    ll i=1,k=2;
    ll x0=rand()%x;
    ll y=x0;
    while(1){
        i++;
        x0=(mul(x0,x0,x)+c)%x;
        ll d=__gcd(y-x0,x);
        if(d!=1&&d!=x) return d;
        if(y==x0) return x;
        if(i==k){y=x0;k+=k;}
    }
}

void debug(){
```

```
    ll n;
    while(cin>>n){
        //if(miller_rabin(n))cout<<"YES\n";
        //else cout<<"NO\n";
        cout<<Pollard_rho(n, rand()%(n-1)+1)<<endl;
    }
}

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



//AC自动机
//查询模式串出现次数
#include <bits/stdc++.h>
using namespace std;
struct Trie{
    int ch[50500][27];
    int fail[50500];
    int end[50500];
    int sz;
    int newnode(){
        for(int i=0;i<27;i++)ch[sz][i]=-1;
        end[sz]=-1;
        return sz++;
    }
    void init(){
        sz=0;
        newnode();
    }
    int idx(int c){
        if(c>='A'&&c<='Z')return c-'A';
        return 26;
    }
    void insert(char s[], int id){
        int len=strlen(s);
        int u=0;
        for(int i=0;i<len;i++){
            int c=idx(s[i]);
            if(ch[u][c]==-1){
                ch[u][c]=newnode();
            }
            u=ch[u][c];
        }
        end[u]=id;
    }
    void build(){
        queue<int>q;
        fail[0]=0;
        for(int i=0;i<27;i++){
            if(ch[0][i]==-1)ch[0][i]=0;
            else {
                fail[ch[0][i]]=0;
                q.push(ch[0][i]);
            }
        }
        while(!q.empty()){
            int u=q.front();
            q.pop();
            for(int i=0;i<27;i++){
                if(ch[u][i]==-1){
                    ch[u][i]=ch[fail[u]][i];
                }
                else {
                    fail[ch[u][i]]=ch[fail[u]][i];
                    q.push(ch[u][i]);
                }
            }
        }
    }
```

```
};

Trie T;
char s[1050][55];
char str[2050000];
int cnt[1050];
int n;

int main()
{
    while(~scanf("%d", &n)){
        memset(cnt, 0, sizeof(cnt));
        T.init();
        for(int i=1;i<=n;i++){
            scanf("%s", s[i]);
            T.insert(s[i], i);
        }
        T.build();
        scanf("%s", str);
        int len=strlen(str);
        int u=0;
        for(int i=0;i<len;i++){
            int c=T.idx(str[i]);
            u=T.ch[u][c];
            int tmp=u;
            while(tmp){
                if(T.end[tmp]!=-1){
                    cnt[T.end[tmp]]++;
                }
                tmp=T.fail[tmp];
            }
        }
        for(int i=1;i<=n;i++){
            if(cnt[i]){
                printf("%s: %d\n", s[i], cnt[i]);
            }
        }
    }
}

//数位DP
//    pos    = 当前处理的位置(一般从高位到低位)
//    pre    = 上一个位的数字(更高的那一位)
//    status = 要达到的状态,如果为1则可以认为找到了答案,到时候用来返回,
//             给计数器+1。
//    limit  = 是否受限,也即当前处理这位能否随便取值。如567,当前处理6这位,
//             如果前面取的是4,则当前这位可以取0-9。如果前面取的5,那么当前
//             这位就不能随便取，不然会超出这个数的范围,所以如果前面取5的
//             话此时的limit=1,也就是说当前只可以取0-6。
//
//    用DP数组保存这三个状态是因为往后转移的时候会遇到很多重复的情况。
int    dfs(int pos,int pre,int status,int limit)
{
    //已结搜到尽头,返回"是否找到了答案"这个状态。
    if(pos < 1)
        return    status;

    //DP里保存的是完整的,也即不受限的答案,所以如果满足的话,可以直接返回。
    if(!limit && DP[pos][pre][status] != -1)
        return    DP[pos][pre][status];

    int    end = limit ? DIG[pos] : 9;
    int    ret = 0;

    //往下搜的状态表示的很巧妙,status用||是因为如果前面找到了答案那么后面
    //还有没有答案都无所谓了。而limti用&&是因为只有前面受限、当前受限才能
    //推出下一步也受限， 比如567,如果是46X的情况,虽然6已经到尽头,但是后面的
    //个位仍然可以随便取,因为百位没受限,所以如果个位要受限,那么前面必须是56。
    //
    //这里用"不要49"一题来做例子。
    for(int i = 0;i <= end;i ++)
        ret += dfs(pos - 1,i,status || (pre == 4 && i == 9),limit && (i == end));

```

```
    //DP里保存完整的、取到尽头的数据
    if(!limit)
        DP[pos][pre][status] = ret;

    return    ret;
}

//点分治
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;
typedef long long ll;

struct Edge{
    int to, next, len;
}edge[20500];

int head[20500];
bool vis[20500];
int siz[20500];
int f[20500];
int cnt, rt, sum, n, k;
ll ans;
ll vs[20500];
int vsc;
ll v[20500];
int vc;

void init(){
    memset(head, -1, sizeof(head));
    memset(vis, false, sizeof(vis));
    cnt=0;
    ans=0;
}

void add(int u, int v, int w){
    edge[cnt].to=v;
    edge[cnt].len=w;
    edge[cnt].next=head[u];
    head[u]=cnt++;
}

void getrt(int u, int fa){
    siz[u]=1;
    f[u]=0;
    for(int i=head[u];~i;i=edge[i].next){
        if(edge[i].to!=fa&&!vis[edge[i].to]){
            getrt(edge[i].to, u);
            siz[u]+=siz[edge[i].to];
            f[u]=max(f[u], siz[edge[i].to]);
        }
    }
    f[u]=max(f[u], sum-siz[u]);
    if(f[u]<f[rt])rt=u;
}

void getdeep(int u, int fa, ll len){
    vs[++vsc]=len;
    v[++vc]=len;
    for(int i=head[u];~i;i=edge[i].next){
        if(edge[i].to!=fa&&!vis[edge[i].to]){
            getdeep(edge[i].to, u, len+edge[i].len);
        }
    }
}

void solve(int u){
    vis[u]=true;
    vsc=0;
    for(int i=head[u];~i;i=edge[i].next){
        if(!vis[edge[i].to]){
```

```
            vc=0;
            getdeep(edge[i].to, u, edge[i].len);
            sort(v+1, v+1+vc);
            int l=1, r=vc;
            while(l<r){
                if(v[l]+v[r]<=k){
                    ans-=(r-l);
                    l++;
                }
                else r--;
            }
        }
    }
    sort(vs+1, vs+1+vsc);
    int l=1, r=vsc;
    while(l<r){
        if(vs[l]+vs[r]<=k){
            ans+=(r-l);
            l++;
        }
        else r--;
    }
    for(int i=1;i<=vsc;i++){
        if(vs[i]<=k)ans++;
    }
    for(int i=head[u];~i;i=edge[i].next){
        if(!vis[edge[i].to]){
            rt=0, sum=siz[edge[i].to];
            getrt(edge[i].to, 0);
            solve(rt);
        }
    }
}

int main()
{
    while(~scanf("%d%d", &n, &k)){
        if(!n&&!k)break;
        init();
        for(int i=1;i<n;i++){
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            add(u, v, w);
            add(v, u, w);
        }
        f[0]=30000;
        rt=0, sum=n;
        getrt(1, 0);
        solve(rt);
        printf("%lld\n", ans);
    }
}

//线性递推模板m^2logn
#include <bits/stdc++.h>
#pragma comment(linker, "/STACK:102400000,102400000")
using namespace std;
typedef long long ll;
#define LL long long
const int mod=1e9+7;
const int MOD=1e9+7;
const int MAXN=205;
ll n;
int u, d;
int a[100], b[100];
ll dp[205];
ll A[205];
ll C[205];

// given first m a[i] and coef (0 based)
// calc a[n] % MOD in O(m*m*log(n))
// a[n] = sum(c[m - i] * a[n - i]) i = 1....m
// a[m] = sum(c[i] * a[i]), i = 0.....m-1
```

```
LL linear_recurrence(LL n, int m, LL a[], LL c[], int p){ //n->a[i], m -> c[i]
    LL v[MAXN] = {1 % MOD}, u[MAXN << 1], msk = !!n;
    for(LL i = n; i > 1; i >>= 1) msk <<= 1;
    for(LL x = 0; msk; msk >>= 1, x <<= 1){
        fill_n(u, m << 1, 0);
        int b = !!(n & msk); x |= b;
        if(x < m) u[x] = 1 % p;
        else{
            for(int i = 0; i < m; i++){
                for(int j = 0, t = i + b; j < m; ++j, ++t)
                    u[t] = (u[t] + v[i] * v[j]) % MOD;
            }
            for(int i = (m << 1) - 1; i >= m; --i){
                for(int j = 0, t = i - m; j < m; ++j, ++t){
                    u[t] = (u[t] + c[j] * u[i]) % MOD;
                }
            }
        }
        copy(u, u+m, v);
    }
    LL ans = 0;
    for(int i = 0; i < m; i++){
        ans = (ans + v[i] * a[i]) % MOD;
    }
    return ans;
}

bool vis[205];
int main()
{
    while(~scanf("%lld", &n)){
        int ma=0;
        memset(dp, 0, sizeof(dp));
        memset(A, 0, sizeof(A));
        memset(C, 0, sizeof(C));
        memset(vis, false, sizeof(vis));
        scanf("%d", &u);
        for(int i=1;i<=u;i++){scanf("%d", &a[i]);}
        scanf("%d", &d);
        for(int i=1;i<=d;i++){scanf("%d", &b[i]);ma=max(b[i], ma);vis[b[i]]=true;}
        dp[0]=1;
        for(int i=1;i<=ma;i++){
            for(int j=1;j<=u;j++){
                if(i<a[j])continue;
                dp[i]=(dp[i-a[j]]+dp[i])%mod;
            }
        }
        vis[0]=true;
        for(int i=0;i<=ma;i++)if(!vis[i])dp[i]=0;
        for(int i=0;i<ma;i++){C[i]=dp[ma-i];}
        A[0]=1;
        for(int i=1;i<ma;i++){
            for(int j=1;j<=i;j++){
                A[i]=(A[i]+A[i-j]*dp[j])%mod;
            }
        }
        ll ans=linear_recurrence(n, ma, A, C, mod);
        printf("%lld\n", ans);
    }
}

//吉司机线段树
void update(int u, int ql, int qr, int c, int l, int r){
    if(r<ql||qr<l||cut())return;
    if(ql<=l&&r<=qr&&check()){
        putlazy(u,c);
        return;
    }
    int mid=(l+r)/2;
    pushdown(u);
    update(2*u, ql, qr, c, l, mid);
    update(2*u+1, ql, qr, c, mid+1, r);
```

```
    pushup(u);
}

//对于hdu5306这个题，考虑区间与t取min这个操作，有如下几种情况：
//case 1：t大于最大值，此时区间不变；
//case 2：t小于严格次大值，此时至少把最大值和次大值变得相同，即使得区间变得相同，允许暴力更新；
//case 3：t大于严格次大值，小于最大值，这里可以打懒标记。
//
//考虑查询，只需维护最大值，最大值个数，严格次大值即可。
//吉司机宇宙线段树之王！
#include <bits/stdc++.h>
using namespace std;
const int maxn=1000005;
typedef long long ll;

int mx[maxn<<2];
int cnt[maxn<<2];
int se[maxn<<2];
int lazy[maxn<<2];
ll sum[maxn<<2];
int a[maxn];
int n, m;

void putlazy(int u, int t){
    sum[u]-=1LL*cnt[u]*(mx[u]-t);
    mx[u]=t;
    lazy[u]=t;
}

void pushdown(int u){
    if(lazy[u]==-1)return;
    if(mx[2*u]>lazy[u]){
        sum[2*u]-=1LL*cnt[2*u]*(mx[2*u]-lazy[u]);
        mx[2*u]=lazy[u];
        lazy[2*u]=lazy[u];
    }
    if(mx[2*u+1]>lazy[u]){
        sum[2*u+1]-=1LL*cnt[2*u+1]*(mx[2*u+1]-lazy[u]);
        mx[2*u+1]=lazy[u];
        lazy[2*u+1]=lazy[u];

    }
    lazy[u]=-1;
}

void pushup(int u){
    if(mx[2*u]==mx[2*u+1]){
        mx[u]=mx[2*u];
        cnt[u]=cnt[2*u]+cnt[2*u+1];
        se[u]=max(se[2*u], se[2*u+1]);
        sum[u]=sum[2*u]+sum[2*u+1];
    }
    else if(mx[2*u]>mx[2*u+1]){
        mx[u]=mx[2*u];
        cnt[u]=cnt[2*u];
        se[u]=max(se[2*u], mx[2*u+1]);
        sum[u]=sum[2*u]+sum[2*u+1];
    }
    else {
        mx[u]=mx[2*u+1];
        cnt[u]=cnt[2*u+1];
        se[u]=max(mx[2*u], se[2*u+1]);
        sum[u]=sum[2*u]+sum[2*u+1];
    }
}

void build(int u, int l, int r){
    lazy[u]=-1;
    if(l==r){
        mx[u]=sum[u]=a[l];
        cnt[u]=1;
```

```
        se[u]=-1;
        return;
    }
    int mid=l+r>>1;
    build(2*u, l, mid);
    build(2*u+1, mid+1, r);
    pushup(u);
}

void update(int u, int ql, int qr, int t, int l, int r){
    if(ql>r||qr<l||mx[u]<=t)return;
    if(ql<=l&&r<=qr&&se[u]<t){
        putlazy(u, t);
        return;
    }
    pushdown(u);
    int mid=l+r>>1;
    update(2*u, ql, qr, t, l, mid);
    update(2*u+1, ql, qr, t, mid+1, r);
    pushup(u);
}

int getmx(int u, int ql, int qr, int l, int r){
    if(ql>r||qr<l)return 0;
    if(ql<=l&&r<=qr)return mx[u];
    pushdown(u);
    int mid=l+r>>1;
    int ans=0;
    ans=max(ans, getmx(2*u, ql, qr, l, mid));
    ans=max(ans, getmx(2*u+1, ql, qr, mid+1, r));
    return ans;
}

ll getsum(int u, int ql, int qr, int l, int r){
    if(ql>r||qr<l)return 0;
    if(ql<=l&&r<=qr)return sum[u];
    pushdown(u);
    int mid=l+r>>1;
    ll ans=0;
    ans+=getsum(2*u, ql, qr, l, mid);
    ans+=getsum(2*u+1, ql, qr, mid+1, r);
    return ans;
}

int main(){
    int T;
    scanf("%d", &T);
    while(T--){
        scanf("%d%d", &n, &m);
        for(int i=1;i<=n;i++)scanf("%d", &a[i]);
        build(1, 1, n);
        for(int i=1;i<=m;i++){
            int tag;
            scanf("%d", &tag);
            if(tag==0){
                int x, y, t;
                scanf("%d%d%d", &x, &y, &t);
                update(1, x, y, t, 1, n);
            }
            else if(tag==1){
                int x, y;
                scanf("%d%d", &x, &y);
                printf("%d\n", getmx(1, x, y, 1, n));
            }
            else {
                int x, y;
                scanf("%d%d", &x, &y);
                printf("%lld\n", getsum(1, x, y, 1, n));
            }
        }
    }
}
```

```cpp
//树链剖分
//给你一颗树，现在有两个操作，一种是改变某条边的权值，一种是查询点u到v之间的路径的最大边权
//input:
//1
//3
//1 2 1
//2 3 2
//QUERY 1 2
//CHANGE 1 3
//QUERY 1 2
//DONE
//
//Output:
//1
//3

#include <bits/stdc++.h>
using namespace std;
const int maxn=10500;

int dep[maxn],siz[maxn],fa[maxn],id[maxn],son[maxn],val[maxn],top[maxn];
int num,n;
int ma[maxn<<2];
vector<int>g[maxn];
struct edge{
    int x,y,val;
    void read(){
        scanf("%d%d%d",&x,&y,&val);
    }
}e[maxn];

void init(){
    for(int i=1;i<=n;i++)g[i].clear();
}

void dfs1(int u,int f,int d){
    dep[u]=d,siz[u]=1,son[u]=0,fa[u]=f;
    for(int i=0;i<g[u].size();i++){
        int v=g[u][i];
        if(v==f)continue;
        dfs1(v,u,d+1);
        siz[u]+=siz[v];
        if(siz[son[u]]<siz[v])son[u]=v;
    }
}

void dfs2(int u,int tp){
    top[u]=tp;
    id[u]=++num;
    if(son[u])dfs2(son[u],tp);
    for(int i=0;i<g[u].size();i++){
        int v=g[u][i];
        if(v==fa[u]||v==son[u])continue;
        dfs2(v,v);
    }
}

void pushup(int u){
    ma[u]=max(ma[2*u],ma[2*u+1]);
}

void build(int u,int l,int r){
    if(l==r){
        ma[u]=val[l];
        return ;
    }
    int mid=(l+r)/2;
    build(2*u,l,mid);
    build(2*u+1,mid+1,r);
    pushup(u);
}
```

```cpp
void update(int u,int p,int l,int r,int q){
    if(p<l||p>r)return;
    if(l==r){
        ma[u]=q;
        return;
    }
    int mid=(l+r)/2;
    update(2*u,p,l,mid,q);
    update(2*u+1,p,mid+1,r,q);
    pushup(u);
}

int query(int u,int ql,int qr,int l,int r){
    if(ql>r||qr<l)return 0;
    if(ql<=l&&r<=qr)return ma[u];
    int mid=(l+r)/2;
    return max(query(2*u,ql,qr,l,mid),query(2*u+1,ql,qr,mid+1,r));
}


int Query(int u,int v){
    int ret=0;
    int tp1=top[u],tp2=top[v];
    while(tp1!=tp2){
        if(dep[tp1]<dep[tp2]){
            swap(u,v);swap(tp1,tp2);
        }
        ret=max(ret,query(1,id[tp1],id[u],1,num));
        u=fa[tp1];
        tp1=top[u];
    }
    if(u==v)return ret;
    if(dep[u]<dep[v])swap(u,v);
    ret=max(ret,query(1,id[son[v]],id[u],1,num));
    return ret;
}




int main(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%d",&n);
        init();
        for(int i=1;i<n;i++){
            e[i].read();
            g[e[i].x].push_back(e[i].y);
            g[e[i].y].push_back(e[i].x);
        }
        num=0;
        dfs1(1,0,1);
        dfs2(1,1);
        for(int i=1;i<n;i++){
            if(dep[e[i].x]<dep[e[i].y])swap(e[i].x,e[i].y);
            val[id[e[i].x]]=e[i].val;
        }
        build(1,1,num);
        char tag[30];
        while(scanf("%s",tag)){
            if(tag[0]=='D')break;
            if(tag[0]=='Q'){
                int x,y;
                scanf("%d%d",&x,&y);
                printf("%d\n",Query(x,y));
            }
            else {
                int pos,x;
                scanf("%d%d",&pos,&x);
                update(1,id[e[pos].x],1,num,x);
            }
        }
```

```
    }
}

//二分图匹配
//用和为素数的二元组覆盖最多的数
#include <bits/stdc++.h>
using namespace std;
const int maxn=3003;
const int maxe=3003*3003;
vector<int>g[maxn];
int head[maxn], linker[maxn], vis[maxn], n, k, sum;
void init(){
    memset(linker, -1, sizeof(linker));
    for(int i=0;i<n;i++)g[i].clear();
    sum=0;
}
void add(int u, int v){
    g[u].push_back(v);
}
int num[maxn];
bool isprime[2000005];
void init1(){
    memset(isprime, true, sizeof(isprime));
    isprime[0]=isprime[1]=false;
    for(int i=2;i<=2000000;i++){
        if(isprime[i]){
            for(int j=2;j*i<=2000000;j++)isprime[i*j]=false;
        }
    }
}
bool cmp(int x, int y){
    return x>y;
}
int Find(int u){
    for(int i=0;i<g[u].size();i++){
        int v=g[u][i];
        if(!vis[v]){
            vis[v]=1;
            if(linker[v]==-1||Find(linker[v])){
                linker[v]=u;
                linker[u]=v;
                return 1;
            }
        }
    }
    return 0;
}
int match(){
    int ans=0;
    for(int i=0;i<n;i++){
        if(num[i]&1&&linker[i]==-1){
            memset(vis, 0, sizeof(vis));
            vis[i]=true;
            ans+=Find(i);
        }
    }
    return ans;
}
int main(){
    init1();
    int T;
    scanf("%d", &T);
    while(T--){
        scanf("%d%d", &n, &k);
        init();
        for(int i=0;i<n;i++){
            scanf("%d", &num[i]);
        }
        sort(num, num+n, cmp);
        for(int i=0;i<n;i++){
            bool ok=false;
            for(int j=0;j<n;j++){
                if(i==j)continue;
```

```
                if(isprime[num[i]+num[j]]){
                    ok=true;
                    add(i, j);
                }
            }
            if(ok)sum++;
        }
        for(int i=0;i<n;i++)sort(g[i].begin(), g[i].end());
        if(sum<=k){
            printf("%d\n", sum);
        }
        else {
            int m=match();
            if(m>=k)printf("%d\n", 2*k);
            else if(sum-m<=k)printf("%d\n", sum);
            else printf("%d\n", k+m);
        }
    }
}

//bzoj1010
//决策单调性
//
//P教授要去看奥运，但是他舍不下他的玩具，于是他决定把所有的玩具运到北京。他使用自己的压缩器进行压缩，其可以将任意物品变成一堆，再放到一种特殊的一维容器中。P教授有编号为1…N的N件玩具，第i件玩具经过压缩后变成一维长度为Ci.为了方便整理，P教授要求在一个一维容器中的玩具编号是连续的。同时如果一个一维容器中有多个玩具，那么两件玩具之间要加入一个单位长度的填充物，形式地说如果将第i件玩具到第j个玩具放到一个容器中，那么容器的长度将为x=j-i+Sigma(Ck) i<=K<=j
制作容器的费用与容器的长度有关，根据教授研究，如果容器长度为x,其制作费用为(X-L)^2.其中L是一个常量。P教授不关心容器的数目，他可以制作出任意长度的容器，甚至超过L。但他希望费用最小.
//
//input
//5 4
//3
//4
//2
//1
//4
//
//output
//1
//
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=50005;
ll a[maxn], s[maxn], dp[maxn], l;
int n;
struct node{
    int l, r, p;
}q[maxn];
ll cal(int j, int i){
    return dp[j]+(s[i]-s[j]+i-j-1-l)*(s[i]-s[j]+i-j-1-l);
}
int bisearch(node a, int i){
    int l=a.l, r=a.r+1, tag=0;//这里我的二分默认在l到r之间有答案的,而实际上有可能a.r并不满足，所以把区间右端点+1
    while(l<=r){
        if(r-l<=1){
            if(cal(i, l)<cal(a.p, l))tag=l;
            else tag=r;
            break;
        }
        int mid=l+r>>1;
        if(cal(i, mid)<cal(a.p, mid))r=mid;
        else l=mid;
    }
    return tag;
}
void solve(){
    int head=1, tail=0;
```

```
    q[++tail]=(node){0, n, 0};
    for(int i=1;i<=n;i++){
        if(i>q[head].r)head++;
        dp[i]=cal(q[head].p, i);
        /*for(int i=head;i<=tail;i++){
                printf("node %d %d %d\n", q[i].l, q[i].r, q[i].p);
            }
        cout<<i<<' '<<q[head].p<<' '<<dp[i]<<endl;*/
        if(head>tail||cal(i, n)<cal(q[tail].p, n)){
            while(head<=tail&&cal(i, q[tail].l)<cal(q[tail].p, q[tail].l))tail--;
            if(head<=tail){
                int t=bisearch(q[tail], i);
                q[tail].r=t-1;
                q[++tail]=(node){t, n, i};
            }
            else q[++tail]=(node){i, n, i};
        }
    }
}
int main(){
    scanf("%d%lld", &n, &l);
    for(ll i=1;i<=n;i++){
        scanf("%lld", &a[i]);
        s[i]=s[i-1]+a[i];
    }
    solve();
    printf("%lld\n", dp[n]);
}



//#include <bitset>
//using std::bitset;
//bitset<n>b;
//b.any() any 1?
//b.none() no 1?
//b.count() number of 1?
//b.size() size?
//b.set() set all pos 1
//b.reset() set all pos 0
//b.flip() 1->0, 0->1

//k中浓度为ai/1000的液体配成n/1000的浓度至少需要多少升
//每种溶液均为整数升
//易知若有解必小于1000，dp过程用bitset加速

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

int n, k;
bool a[1050];
bitset<2050>b[2];

int main(){
    scanf("%d%d", &n, &k);
    for(int i=1;i<=k;i++){
        int x;
        scanf("%d", &x);
        a[x]=true;
    }
    b[0][1000]=1;
    for(int i=0;i<=1000;i++){
        if(i!=0&&b[i%2][1000]){
            printf("%d\n", i);
            return 0;
        }
        int now=i%2;
        b[now^1].reset();
        for(int j=0;j<=1000;j++){
            if(a[j]){
                b[now^1]|=(b[now]<<j)>>n;
            }
        }
```

```
    }
    printf("-1\n");
}



//好题
//问题归结为:
//能否将总重量不大于1e6的砝码分成两堆, 其中一堆重量为k
//用多重背包的按位拆分+bitset可以达到o(k*sigma(logAi)/64)
//而由于sigma(i*Ai)=n, 最坏情况O(sigma(logAi))~O(sqrt(n))
//所以总复杂度是O(k*sqrt(n)/64)
#include <bits/stdc++.h>
using namespace std;

int a[1050000];
bool vis[1050000];
int cnt[1050000];
bitset<1050000>b;
int n, k;
int ma=0, mi, bonus=0;

int main(){
    scanf("%d%d", &n, &k);
    for(int i=1;i<=n;i++)scanf("%d", &a[i]);
    for(int i=1;i<=n;i++){
        if(vis[i])continue;
        int tmp=i;
        int len=0;
        while(!vis[tmp]){
            vis[tmp]=true;
            len++;
            tmp=a[tmp];
        }
        cnt[len]++;
        bonus+=len/2;
    }
    ma=min(n, k+min(bonus, k));
    b[0]=1;
    for(int i=1;i<=n;i++){
        if(!cnt[i])continue;
        int k=1;
        while(k<cnt[i]){
            b|=b<<k*i;
            cnt[i]-=k;
            k<<=1;
        }
        b|=b<<cnt[i]*i;
    }
    if(b[k])mi=k;
    else mi=k+1;
    printf("%d %d\n", mi, ma);
}

//树状数组区间修改区间查询
//sum[n]=sigma(a[i])+sigma((n+1-i)d[i]);
//其中a[i]是原数组元素
//d[i]是[i, n]的共同增量
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll a[205000];
ll d[205000];
ll di[205000];
int n, q;

int lowbit(int x){
    return x&(-x);
}

void update(int x, int num){
    ll add=1LL*num;
    ll addi=1LL*num*x;
```

```
    while(x<=n){
        d[x]+=add;
        di[x]+=addi;
        x+=lowbit(x);
    }
}

ll query(int x){
    ll ans=a[x];
    ll tmp=x;
    while(tmp){
        ans+=d[tmp]*(x+1);
        ans+=di[tmp]*(-1);
        tmp-=lowbit(tmp);
    }
    return ans;
}

int main()
{
    scanf("%d", &n);
    for(int i=1;i<=n;i++){
        scanf("%lld", &a[i]);
        a[i]+=a[i-1];
    }
    scanf("%d", &q);
    while(q--){
        int tag;
        scanf("%d", &tag);
        if(tag==1){
            int x, y, num;
            scanf("%d%d%d", &x, &y, &num);
            update(x, num);
            update(y+1, -num);
        }
        else {
            int x, y;
            scanf("%d%d", &x, &y);
            printf("%lld\n", query(y)-query(x-1));
        }
    }
}

//SA
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
const int maxn=20500;
int s[maxn];
int sa[maxn], t[maxn], t2[maxn], c[maxn], n;
int Rank[maxn], height[maxn];
void build_sa(int m, int n){
    int i, *x=t, *y=t2;
    for(i=0;i<m;i++)c[i]=0;
    for(i=0;i<n;i++)c[x[i]=s[i]]++;
    for(i=1;i<m;i++)c[i]+=c[i-1];
    for(i=n-1;i>=0;i--)sa[--c[x[i]]]=i;
    for(int k=1;k<=n;k<<=1){
        int p=0;
        for(i=n-k;i<n;i++)y[p++]=i;
        for(i=0;i<n;i++)if(sa[i]>=k)y[p++]=sa[i]-k;
        for(i=0;i<m;i++)c[i]=0;
        for(i=0;i<n;i++)c[x[y[i]]]++;
        for(i=1;i<m;i++)c[i]+=c[i-1];
        for(i=n-1;i>=0;i--)sa[--c[x[y[i]]]]=y[i];
        swap(x, y);
        p=1;x[sa[0]]=0;
        for(i=1;i<n;i++)
            x[sa[i]]=y[sa[i-1]]==y[sa[i]]&&y[sa[i-1]+k]==y[sa[i]+k]?p-1:p++;
        if(p>=n)break;
        m=p;
    }
```

```
}
void geth(int n){
    int i, j, k=0;
    for(i=0;i<n;i++)Rank[sa[i]]=i;
    for(i=0;i<n;i++){
        if(k)k--;
        if(Rank[i]==0)break;
        j=sa[Rank[i]-1];
        while(s[i+k]==s[j+k])k++;
        height[Rank[i]]=k;
    }
}

int main()
{
    n=8;
    s[8]=0;
    s[0]=s[1]=2;
    s[2]=s[3]=s[4]=s[5]=1;
    s[6]=s[7]=3;
    build_sa(200, n+1);
    geth(n+1);
    for(int i=0;i<=n;i++){
        printf("%d ", sa[i]);
    }
    printf("\n");
    for(int i=1;i<=n;i++){
        printf("%d ", height[i]);
    }
    printf("\n");

}
```