```
/* 数学 */
/* 扩展欧几里得 */
/* 素数线性筛 */
/* 中国剩余定理 */
/* 卢卡斯定理 */
/* 逆元 */
/* 欧拉函数 */
/* 莫比乌斯函数 */
/* 威佐夫博弈 */
/* 反NIM博弈 */
/* NTT/FFT/FWT */
/* Miller_Rabin */
/* 线性递推m^2logn */
/* polya定理 */

/* 数据结构 */
/* 单调队列 */
/* 吉司机线段树 */
/* 主席树 */
/* splay */

/* 字符串 */
/* 后缀数组 */
/* AC自动机 */

/* 图论 */
/* lca */
/* tarjan */
/* 2-sat */
/* 网络流 */
/* KM */
/* 欧拉图 */
/* 点分治 */

/* dp */
/* 数位dp */
/* 斜率优化 */
/* 四边形优化 */

/* 其他 */
/* 输入外挂 */
/* bitset */
/* O(1)快速乘 */

//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;
    complex(double _r = 0.0,double _i = 0.0)
```

```
    {
        r = _r; i = _i;
    }
    complex operator +(const complex &b)
    {
        return complex(r+b.r,i+b.i);
    }
    complex operator -(const complex &b)
    {
        return complex(r-b.r,i-b.i);
    }
    complex operator *(const complex &b)
    {
        return complex(r*b.r-i*b.i,r*b.i+i*b.r);
    }
};
/*
 * 进行FFT和IFFT前的反转变换。
 * 位置i和 （i二进制反转后位置）互换
 * len必须去2的幂
 */
void change(complex y[],int len)
{
    int i,j,k;
    for(i = 1, j = len/2;i < len-1; i++)
    {
        if(i < j)swap(y[i],y[j]);
        //交换互为小标反转的元素，i<j保证交换一次
        //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
        k = len/2;
        while( j >= k)
        {
            j -= k;
            k /= 2;
        }
        if(j < k) j += k;
    }
}
/*
 * 做FFT
 * len必须为2^k形式，
 * on==1时是DFT，on==-1时是IDFT
 */
void fft(complex y[],int len,int on)
{
    change(y,len);
    for(int h = 2; h <= len; h <<= 1)
    {
        complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
        for(int j = 0;j < len;j+=h)
        {
            complex w(1,0);
            for(int k = j;k < j+h/2;k++)
            {
                complex u = y[k];
                complex t = w*y[k+h/2];
                y[k] = u+t;
                y[k+h/2] = u-t;
                w = w*wn;
            }
        }
    }
    if(on == -1)
        for(int i = 0;i < len;i++)
            y[i].r /= len;
}
const int MAXN = 200010;
complex x1[MAXN],x2[MAXN];
char str1[MAXN/2],str2[MAXN/2];
int sum[MAXN];
int main()
{
    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=5000000001507329LL; // 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);
    }
}
```

```cpp
int len;
void NTT(long long y[],int op)
{
    for(int i=1,j=len>>1,k; i<len-1; ++i)
    {
        if(i<j) swap(y[i],y[j]);
        k=len>>1;
        while(j>=k)
        {
            j-=k;
            k>>=1;
        }
        if(j<k) j+=k;
    }
    int id=0;
    for(int h=2; h<=len; h<<=1)
    {
        ++id;
        for(int i=0; i<len; i+=h)
        {
            long long w=1;
            for(int j=i; j<i+(h>>1); ++j)
            {
                long long u=y[j],t=mul(y[j+h/2],w);
                y[j]=u+t;
                if(y[j]>=P) y[j]-=P;
                y[j+h/2]=u-t+P;
                if(y[j+h/2]>=P) y[j+h/2]-=P;
                w=mul(w,wn[id]);
            }
        }
    }
    if(op==-1)
    {
        for(int i=1; i<len/2; ++i) swap(y[i],y[len-i]);
        long long inv=qpow(len,P-2,P);
        for(int i=0; i<len; ++i) y[i]=mul(y[i],inv);
    }
}
void Convolution(long long A[],long long B[],int len1,int len2)
{
    int n=max(len1,len2);
    for(len=1; len<(n<<1); len<<=1);
    for(int i=len1; i<len; ++i)
    {
        A[i]=0;
    }
    for (int i=len2;i<len;i++)
    {
        B[i]=0;
    }

    NTT(A,1);
    NTT(B,1);
    for(int i=0; i<len; ++i)
    {
        A[i]=mul(A[i],B[i]);
    }
    NTT(A,-1);
}

long long A[MAXN],B[MAXN];
char s1[MAXN],s2[MAXN];
void debug(){
    A[0]=1, A[1]=2, A[2]=3;
    B[0]=1, B[1]=2, B[2]=3;
    Convolution(A, B, 3, 3);
    for(int i=0;i<=6;i++)printf("%lld\n", A[i]);
}
int main()
{
    getwn();
    debug();
```

```
}

//FWT
//rm to do

//Miller Rabin

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

typedef unsigned long long ll;

const int T=10;

ll exp(ll x, ll y, ll mod){
    ll ans=1;
    ll base=x;
    while(y){
        if(y&1){
            ans=ans*base%mod;
        }
        y>>=1;
        base=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=x*x%n;
            if(y==1&&x!=1&&x!=n-1)return false;
            x=y;
        }
        if(y!=1)return false;
    }
    return true;
}

void debug(){
    int n;
    while(cin>>n){
        if(miller_rabin(n))cout<<"YES\n";
        else cout<<"NO\n";
    }
}

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

//后缀数组
#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];
```

```
439 void build_sa(int m, int n){
440     int i, *x=t, *Gy=t2;
441     for(i=0;i<m;i++)c[i]=0;
442     for(i=0;i<n;i++)c[x[i]=s[i]]++;
443     for(i=1;i<m;i++)c[i]+=c[i-1];
444     for(i=n-1;i>=0;i--)sa[--c[x[i]]]=i;
445     for(int k=1;k<=n;k<<=1){
446         int p=0;
447         for(i=n-k;i<n;i++)y[p++]=i;
448         for(i=0;i<n;i++)if(sa[i]>=k)y[p++]=sa[i]-k;
449         for(i=0;i<m;i++)c[i]=0;
450         for(i=0;i<n;i++)c[x[y[i]]]++;
451         for(i=1;i<m;i++)c[i]+=c[i-1];
452         for(i=n-1;i>=0;i--)sa[--c[x[y[i]]]]=y[i];
453         swap(x, y);
454         p=1;x[sa[0]]=0;
455         for(i=1;i<n;i++)
456             x[sa[i]]=y[sa[i-1]]==y[sa[i]]&&y[sa[i-1]+k]==y[sa[i]+k]?p-1:p++;
457         if(p>=n)break;
458         m=p;
459     }
460 }
461 void geth(int n){
462     int i, j, k=0;
463     for(i=0;i<n;i++)Rank[sa[i]]=i;
464     for(i=0;i<n;i++){
465         if(k)k--;
466         if(Rank[i]==0)break;
467         j=sa[Rank[i]-1];
468         while(s[i+k]==s[j+k])k++;
469         height[Rank[i]]=k;
470     }
471 }
472
473 int main()
474 {
475     n=8;
476     s[8]=0;
477     s[0]=s[1]=2;
478     s[2]=s[3]=s[4]=s[5]=1;
479     s[6]=s[7]=3;
480     build_sa(200, n+1);
481     geth(n+1);
482     for(int i=0;i<=n;i++){
483         printf("%d ", sa[i]);
484     }
485     printf("\n");
486     for(int i=1;i<=n;i++){
487         printf("%d ", height[i]);
488     }
489     printf("\n");
490 }
491
492 //AC自动机
493 //查询模式串出现次数
494 #include <bits/stdc++.h>
495 using namespace std;
496 struct Trie{
497     int ch[50500][27];
498     int fail[50500];
499     int end[50500];
500     int sz;
501     int newnode(){
502         for(int i=0;i<27;i++)ch[sz][i]=-1;
503         end[sz]=-1;
504         return sz++;
505     }
506     void init(){
507         sz=0;
508         newnode();
509     }
510     int idx(int c){
511         if(c>='A'&&c<='Z')return c-'A';
```

```
512             return 26;
513         }
514         void insert(char s[], int id){
515             int len=strlen(s);
516             int u=0;
517             for(int i=0;i<len;i++){
518                 int c=idx(s[i]);
519                 if(ch[u][c]==-1){
520                     ch[u][c]=newnode();
521                 }
522                 u=ch[u][c];
523             }
524             end[u]=id;
525         }
526         void build(){
527             queue<int>q;
528             fail[0]=0;
529             for(int i=0;i<27;i++){
530                 if(ch[0][i]==-1)ch[0][i]=0;
531                 else {
532                     fail[ch[0][i]]=0;
533                     q.push(ch[0][i]);
534                 }
535             }
536             while(!q.empty()){
537                 int u=q.front();
538                 q.pop();
539                 for(int i=0;i<27;i++){
540                     if(ch[u][i]==-1){
541                         ch[u][i]=ch[fail[u]][i];
542                     }
543                     else {
544                         fail[ch[u][i]]=ch[fail[u]][i];
545                         q.push(ch[u][i]);
546                     }
547                 }
548             }
549         }
550     };
551
552     Trie T;
553     char s[1050][55];
554     char str[2050000];
555     int cnt[1050];
556     int n;
557
558     int main()
559     {
560         while(~scanf("%d", &n)){
561             memset(cnt, 0, sizeof(cnt));
562             T.init();
563             for(int i=1;i<=n;i++){
564                 scanf("%s", s[i]);
565                 T.insert(s[i], i);
566             }
567             T.build();
568             scanf("%s", str);
569             int len=strlen(str);
570             int u=0;
571             for(int i=0;i<len;i++){
572                 int c=T.idx(str[i]);
573                 u=T.ch[u][c];
574                 int tmp=u;
575                 while(tmp){
576                     if(T.end[tmp]!=-1){
577                         cnt[T.end[tmp]]++;
578                     }
579                     tmp=T.fail[tmp];
580                 }
581             }
582             for(int i=1;i<=n;i++){
583                 if(cnt[i]){
584                     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));
```

```
804             memset(C, 0, sizeof(C));
805             memset(vis, false, sizeof(vis));
806             scanf("%d", &u);
807             for(int i=1;i<=u;i++){scanf("%d", &a[i]);}
808             scanf("%d", &d);
809             for(int i=1;i<=d;i++){scanf("%d", &b[i]);ma=max(b[i], ma);vis[b[i]]=true;}
810             dp[0]=1;
811             for(int i=1;i<=ma;i++){
812                 for(int j=1;j<=u;j++){
813                     if(i<a[j])continue;
814                     dp[i]=(dp[i-a[j]]+dp[i])%mod;
815                 }
816             }
817             vis[0]=true;
818             for(int i=0;i<=ma;i++)if(!vis[i])dp[i]=0;
819             for(int i=0;i<ma;i++){C[i]=dp[ma-i];}
820             A[0]=1;
821             for(int i=1;i<ma;i++){
822                 for(int j=1;j<=i;j++){
823                     A[i]=(A[i]+A[i-j]*dp[j])%mod;
824                 }
825             }
826             ll ans=linear_recurrence(n, ma, A, C, mod);
827             printf("%lld\n", ans);
828         }
829     }
830
831     //吉司机线段树
832     void update(int u, int ql, int qr, int c, int l, int r){
833         if(r<ql||qr<l||cut())return;
834         if(ql<=l&&r<=qr&&check()){
835             putlazy(u,c);
836             return;
837         }
838         int mid=(l+r)/2;
839         pushdown(u);
840         update(2*u, ql, qr, c, l, mid);
841         update(2*u+1, ql, qr, c, mid+1, r);
842         pushup(u);
843     }
844
845     //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：
846     //case 1：t大于最大值，此时区间不变；
847     //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
        ；
848     //case 3：t大于严格次大值，小于最大值，这里可以打懒标记。
849     //
850     //考虑查询，只需维护最大值，最大值个数，严格次大值即可。
851     //吉司机宇宙线段树之王！
852     #include <bits/stdc++.h>
853     using namespace std;
854     const int maxn=1000005;
855     typedef long long ll;
856
857     int mx[maxn<<2];
858     int cnt[maxn<<2];
859     int se[maxn<<2];
860     int lazy[maxn<<2];
861     ll sum[maxn<<2];
862     int a[maxn];
863     int n, m;
864
865     void putlazy(int u, int t){
866         sum[u]-=1LL*cnt[u]*(mx[u]-t);
867         mx[u]=t;
868         lazy[u]=t;
869     }
870
871     void pushdown(int u){
872         if(lazy[u]==-1)return;
873         if(mx[2*u]>lazy[u]){
874             sum[2*u]-=1LL*cnt[2*u]*(mx[2*u]-lazy[u]);
```

```
875             mx[2*u]=lazy[u];
876             lazy[2*u]=lazy[u];
877         }
878         if(mx[2*u+1]>lazy[u]){
879             sum[2*u+1]-=1LL*cnt[2*u+1]*(mx[2*u+1]-lazy[u]);
880             mx[2*u+1]=lazy[u];
881             lazy[2*u+1]=lazy[u];
882 
883         }
884         lazy[u]=-1;
885     }
886 
887     void pushup(int u){
888         if(mx[2*u]==mx[2*u+1]){
889             mx[u]=mx[2*u];
890             cnt[u]=cnt[2*u]+cnt[2*u+1];
891             se[u]=max(se[2*u], se[2*u+1]);
892             sum[u]=sum[2*u]+sum[2*u+1];
893         }
894         else if(mx[2*u]>mx[2*u+1]){
895             mx[u]=mx[2*u];
896             cnt[u]=cnt[2*u];
897             se[u]=max(se[2*u], mx[2*u+1]);
898             sum[u]=sum[2*u]+sum[2*u+1];
899         }
900         else {
901             mx[u]=mx[2*u+1];
902             cnt[u]=cnt[2*u+1];
903             se[u]=max(mx[2*u], se[2*u+1]);
904             sum[u]=sum[2*u]+sum[2*u+1];
905         }
906     }
907 
908     void build(int u, int l, int r){
909         lazy[u]=-1;
910         if(l==r){
911             mx[u]=sum[u]=a[l];
912             cnt[u]=1;
913             se[u]=-1;
914             return;
915         }
916         int mid=l+r>>1;
917         build(2*u, l, mid);
918         build(2*u+1, mid+1, r);
919         pushup(u);
920     }
921 
922     void update(int u, int ql, int qr, int t, int l, int r){
923         if(ql>r||qr<l||mx[u]<=t)return;
924         if(ql<=l&&r<=qr&&se[u]<t){
925             putlazy(u, t);
926             return;
927         }
928         pushdown(u);
929         int mid=l+r>>1;
930         update(2*u, ql, qr, t, l, mid);
931         update(2*u+1, ql, qr, t, mid+1, r);
932         pushup(u);
933     }
934 
935     int getmx(int u, int ql, int qr, int l, int r){
936         if(ql>r||qr<l)return 0;
937         if(ql<=l&&r<=qr)return mx[u];
938         pushdown(u);
939         int mid=l+r>>1;
940         int ans=0;
941         ans=max(ans, getmx(2*u, ql, qr, l, mid));
942         ans=max(ans, getmx(2*u+1, ql, qr, mid+1, r));
943         return ans;
944     }
945 
946     ll getsum(int u, int ql, int qr, int l, int r){
947         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));
            }
        }
    }
}

//#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");
    }

    //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));
            }
        }
    }
```