

```

1 //扩展欧几里得
2 //素数线性筛
3 //中国剩余定理
4 //逆元
5 //FFT
6 //NTT
7 //MILLER RABIN
8 //AC自动机
9 //数位DP
10 //点分治
11 //线性递推模板 $m^{2\log n}$ 
12 //吉司机线段树
13 //树链剖分
14 //二分图匹配
15 //决策单调性
16 //bitset
17 //树状数组区间修改区间查询
18 //SA
19
20 //扩展欧几里得
21 //求 $ax+by=\gcd(a, b)$ 的解
22 void exgcd(ll a, ll b, ll &x, ll &y){
23     if(b==0){
24         x=1, y=0;
25         return ;
26     }
27     exgcd(b, a%b, x, y);
28     ll tmp=x;
29     x=y;
30     y=tmp-(a/b)*y;
31 }
32
33 //素数线性筛
34 const int maxn=205000;
35
36 bool vis[maxn];
37 int prime[maxn];
38 int tot;
39 void init(){
40     memset(vis, false, sizeof(vis));
41     tot=0;
42     for(int i=2;i<maxn;i++){
43         if(!vis[i])prime[++tot]=i;
44         for(int j=1;j<=tot&&prime[j]*i<maxn;j++){
45             vis[prime[j]*i]=true;
46             if(i%prime[j]==0)break;
47         }
48     }
49 }
50
51 //中国剩余定理
52 //正整数 $m_1, m_2, \dots, m_k$ 两两互素, 则同余方程组
53 //
54 // $x \equiv a_1 \pmod{m_1}$ 
55 // $x \equiv a_2 \pmod{m_2}$ 
56 //
57 // $x \equiv a_k \pmod{m_k}$ 
58 //
59 //在模 $M=m_1*m_2*\dots*m_k$ 下的解唯一
60 //
61 // $x \equiv \sum (a_i * M_i * \text{inv}(M_i)) \pmod{M}$ 
62 //
63 // $M_i = M/m_i$ ,  $\text{inv}(M_i)$  为 $M_i$ 模 $m_i$ 的逆元
64 int CRT(int a[], int m[], int n){
65     int M=1;
66     int ans=0;
67     for(int i=1;i<=n;i++)M*=m[i];
68     for(int i=1;i<=n;i++){
69         int x, y;
70         int Mi=M/m[i];
71         exgcd(Mi, m[i], x, y); //m[i]不一定是素数, 故要用exgcd求逆元
72         ans=(ans+a[i]*Mi*x)%M;
73     }

```

```

74     if(ans<0)ans+=M;
75     return ans;
76 }
77
78 //卢卡斯定理
79 //rm to do
80
81 //逆元
82 //ans=(a/b)%m=(a%(mb))/b
83 //fermat小定理
84 //exgcd
85 //o(n)预处理mod的逆元
86 int mod;
87 int inv[maxn];
88
89 void init(){
90     inv[1] = 1;
91     for(int i=2;i<mod;i++){
92         inv[i]=inv[mod%i]*(mod-mod/i)%mod;
93     }
94 }
95
96 //欧拉函数
97 //rm to do
98
99 //莫比乌斯函数
100 //rm to do
101
102 //威佐夫博弈
103 //rm to do
104
105 //反NIM博弈
106 //rm to do
107
108 //FFT
109 #include <stdio.h>
110 #include <string.h>
111 #include <iostream>
112 #include <algorithm>
113 #include <math.h>
114 using namespace std;
115
116 const double PI = acos(-1.0);
117 //复数结构体
118 struct complex
119 {
120     double r,i;
121     complex(double _r = 0.0,double _i = 0.0)
122     {
123         r = _r; i = _i;
124     }
125     complex operator +(const complex &b)
126     {
127         return complex(r+b.r,i+b.i);
128     }
129     complex operator -(const complex &b)
130     {
131         return complex(r-b.r,i-b.i);
132     }
133     complex operator *(const complex &b)
134     {
135         return complex(r*b.r-i*b.i,r*b.i+i*b.r);
136     }
137 };
138 /*
139  * 进行FFT和IFFT前的反转变换。
140  * 位置i和 (i二进制反转后位置) 互换
141  * len必须去2的幂
142  */
143 void change(complex y[],int len)
144 {
145     int i,j,k;
146     for(i = 1, j = len/2; i < len-1; i++)

```

```

147     {
148         if(i < j) swap(y[i], y[j]);
149         //交换互为小标反转的元素, i<j保证交换一次
150         //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
151         k = len/2;
152         while( j >= k)
153         {
154             j -= k;
155             k /= 2;
156         }
157         if(j < k) j += k;
158     }
159 }
160 /*
161  * 做FFT
162  * len必须为2^k形式,
163  * on==1时是DFT, on==-1时是IDFT
164  */
165 void fft(complex y[], int len, int on)
166 {
167     change(y, len);
168     for(int h = 2; h <= len; h <<= 1)
169     {
170         complex wn(cos(-on*2*PI/h), sin(-on*2*PI/h));
171         for(int j = 0; j < len; j+=h)
172         {
173             complex w(1, 0);
174             for(int k = j; k < j+h/2; k++)
175             {
176                 complex u = y[k];
177                 complex t = w*y[k+h/2];
178                 y[k] = u+t;
179                 y[k+h/2] = u-t;
180                 w = w*wn;
181             }
182         }
183     }
184     if(on == -1)
185         for(int i = 0; i < len; i++)
186             y[i].r /= len;
187 }
188 const int MAXN = 200010;
189 complex x1[MAXN], x2[MAXN];
190 char str1[MAXN/2], str2[MAXN/2];
191 int sum[MAXN];
192 int main()
193 {
194     while(scanf("%s%s", str1, str2) == 2)
195     {
196         int len1 = strlen(str1);
197         int len2 = strlen(str2);
198         int len = 1;
199         while(len < len1*2 || len < len2*2) len <<= 1;
200         for(int i = 0; i < len1; i++)
201             x1[i] = complex(str1[len1-1-i] - '0', 0);
202         for(int i = len1; i < len; i++)
203             x1[i] = complex(0, 0);
204         for(int i = 0; i < len2; i++)
205             x2[i] = complex(str2[len2-1-i] - '0', 0);
206         for(int i = len2; i < len; i++)
207             x2[i] = complex(0, 0);
208         //求DFT
209         fft(x1, len, 1);
210         fft(x2, len, 1);
211         for(int i = 0; i < len; i++)
212             x1[i] = x1[i]*x2[i];
213         fft(x1, len, -1);
214         for(int i = 0; i < len; i++)
215             sum[i] = (int)(x1[i].r+0.5);
216         for(int i = 0; i < len; i++)
217         {
218             sum[i+1] += sum[i]/10;
219             sum[i] %= 10;

```

```

220     }
221     len = len1+len2-1;
222     while(sum[len] <= 0 && len > 0) len--;
223     for(int i = len; i >= 0; i--)
224         printf("%c", sum[i]+'0');
225     printf("\n");
226 }
227 return 0;
228 }
229
230 //NTT
231 #include<cstdio>
232 #include<cstring>
233 #include<algorithm>
234 using namespace std;
235 #define MAXN 262144
236
237 const long long P=50000000001507329LL; // 190734863287 * 2 ^ 18 + 1
238 //const int P=1004535809; // 479 * 2 ^ 21 + 1
239 //const int P=998244353; // 119 * 2 ^ 23 + 1
240 const int G=3;
241
242 long long mul(long long x, long long y)
243 {
244     return (x*y-(long long) (x/(long double) P*y+1e-3)*P+P)%P;
245 }
246 long long qpow(long long x, long long k, long long p)
247 {
248     long long ret=1;
249     while(k)
250     {
251         if(k&1) ret=mul(ret,x);
252         k>>=1;
253         x=mul(x,x);
254     }
255     return ret;
256 }
257
258 long long wn[25];
259 void getwn()
260 {
261     for(int i=1; i<=18; ++i)
262     {
263         int t=1<<i;
264         wn[i]=qpow(G, (P-1)/t, P);
265     }
266 }
267
268 int len;
269 void NTT(long long y[], int op)
270 {
271     for(int i=1, j=len>>1, k; i<len-1; ++i)
272     {
273         if(i<j) swap(y[i], y[j]);
274         k=len>>1;
275         while(j>=k)
276         {
277             j-=k;
278             k>>=1;
279         }
280         if(j<k) j+=k;
281     }
282     int id=0;
283     for(int h=2; h<=len; h<<=1)
284     {
285         ++id;
286         for(int i=0; i<len; i+=h)
287         {
288             long long w=1;
289             for(int j=i; j<i+(h>>1); ++j)
290             {
291                 long long u=y[j], t=mul(y[j+h/2], w);
292                 y[j]=u+t;

```

```

293         if(y[j]>=P) y[j]-=P;
294         y[j+h/2]=u-t+P;
295         if(y[j+h/2]>=P) y[j+h/2]-=P;
296         w=mul(w,wn[id]);
297     }
298 }
299 }
300 if(op== -1)
301 {
302     for(int i=1; i<len/2; ++i) swap(y[i],y[len-i]);
303     long long inv=qpow(len,P-2,P);
304     for(int i=0; i<len; ++i) y[i]=mul(y[i],inv);
305 }
306 }
307 void Convolution(long long A[],long long B[],int len1,int len2)
308 {
309     int n=max(len1,len2);
310     for(len=1; len<(n<<1); len<<=1);
311     for(int i=len1; i<len; ++i)
312     {
313         A[i]=0;
314     }
315     for (int i=len2;i<len;i++)
316     {
317         B[i]=0;
318     }
319
320     NTT(A,1);
321     NTT(B,1);
322     for(int i=0; i<len; ++i)
323     {
324         A[i]=mul(A[i],B[i]);
325     }
326     NTT(A,-1);
327 }
328
329 long long A[MAXN],B[MAXN];
330 char s1[MAXN],s2[MAXN];
331 void debug() {
332     A[0]=1, A[1]=2, A[2]=3;
333     B[0]=1, B[1]=2, B[2]=3;
334     Convolution(A, B, 3, 3);
335     for(int i=0;i<=6;i++)printf("%lld\n", A[i]);
336 }
337 int main()
338 {
339     getwn();
340     debug();
341 }
342
343 //FWT
344 //rm to do
345
346 //Miller Rabin
347 //Pollard_rho
348
349 #include <bits/stdc++.h>
350 using namespace std;
351
352 typedef unsigned long long ll;
353
354 const int T=10;
355
356 ll mul(ll x,ll y, ll P)
357 {
358     return (x*y-(ll)(x/(ll)P*y+1e-3)*P+P)%P;
359 }
360
361 ll exp(ll x, ll y, ll mod){
362     ll ans=1;
363     ll base=x;
364     while(y){
365         if(y&1){

```

```

366         ans=mul(ans, base, mod);
367     }
368     y>>=1;
369     base=mul(base, base, mod);
370 }
371 return ans;
372 }
373
374 bool miller_rabin(ll n)
375 {
376     if(n==2) return true;
377     if(n<2||!(n&1)) return false;
378     ll m=n-1;
379     ll k=0;
380     while(!(m&1)){
381         k++;
382         m>>=1;
383     }
384     for(int i=0;i<T;i++){
385         ll a=rand()%(n-1)+1;
386         ll x=exp(a, m, n);
387         ll y=0;
388         for(int j=1;j<=k;j++){
389             y=mul(x, x, n);
390             if(y==1&&x!=1&&x!=n-1) return false;
391             x=y;
392         }
393         if(y!=1) return false;
394     }
395     return true;
396 }
397
398 ll Pollard_rho(ll x,ll c){
399     ll i=1,k=2;
400     ll x0=rand()%x;
401     ll y=x0;
402     while(1){
403         i++;
404         x0=(mul(x0,x0,x)+c)%x;
405         ll d=__gcd(y-x0,x);
406         if(d!=1&&d!=x) return d;
407         if(y==x0) return x;
408         if(i==k){y=x0;k+=k;}
409     }
410 }
411
412 void debug(){
413     ll n;
414     while(cin>>n){
415         //if(miller_rabin(n)) cout<<"YES\n";
416         //else cout<<"NO\n";
417         cout<<Pollard_rho(n, rand()%(n-1)+1)<<endl;
418     }
419 }
420
421 int main(){
422     srand(time(0));
423     debug();
424 }
425
426
427 //AC自动机
428 //查询模式串出现次数
429 #include <bits/stdc++.h>
430 using namespace std;
431 struct Trie{
432     int ch[50500][27];
433     int fail[50500];
434     int end[50500];
435     int sz;
436     int newnode(){
437         for(int i=0;i<27;i++) ch[sz][i]=-1;

```

```

439         end[sz] = -1;
440         return sz++;
441     }
442     void init() {
443         sz = 0;
444         newnode();
445     }
446     int idx(int c) {
447         if(c >= 'A' && c <= 'Z') return c - 'A';
448         return 26;
449     }
450     void insert(char s[], int id) {
451         int len = strlen(s);
452         int u = 0;
453         for(int i = 0; i < len; i++) {
454             int c = idx(s[i]);
455             if(ch[u][c] == -1) {
456                 ch[u][c] = newnode();
457             }
458             u = ch[u][c];
459         }
460         end[u] = id;
461     }
462     void build() {
463         queue<int> q;
464         fail[0] = 0;
465         for(int i = 0; i < 27; i++) {
466             if(ch[0][i] == -1) ch[0][i] = 0;
467             else {
468                 fail[ch[0][i]] = 0;
469                 q.push(ch[0][i]);
470             }
471         }
472         while(!q.empty()) {
473             int u = q.front();
474             q.pop();
475             for(int i = 0; i < 27; i++) {
476                 if(ch[u][i] == -1) {
477                     ch[u][i] = ch[fail[u]][i];
478                 }
479                 else {
480                     fail[ch[u][i]] = ch[fail[u]][i];
481                     q.push(ch[u][i]);
482                 }
483             }
484         }
485     }
486 };
487
488 Trie T;
489 char s[1050][55];
490 char str[2050000];
491 int cnt[1050];
492 int n;
493
494 int main()
495 {
496     while(~scanf("%d", &n)) {
497         memset(cnt, 0, sizeof(cnt));
498         T.init();
499         for(int i = 1; i <= n; i++) {
500             scanf("%s", s[i]);
501             T.insert(s[i], i);
502         }
503         T.build();
504         scanf("%s", str);
505         int len = strlen(str);
506         int u = 0;
507         for(int i = 0; i < len; i++) {
508             int c = T.idx(str[i]);
509             u = T.ch[u][c];
510             int tmp = u;
511             while(tmp) {

```

```

512         if(T.end[tmp] != -1) {
513             cnt[T.end[tmp]]++;
514         }
515         tmp = T.fail[tmp];
516     }
517 }
518 for(int i=1; i<=n; i++) {
519     if(cnt[i]) {
520         printf("%s: %d\n", s[i], cnt[i]);
521     }
522 }
523 }
524 }
525
526 //数位DP
527 // pos = 当前处理的位置(一般从高位到低位)
528 // pre = 上一个位的数字(更高的那一位)
529 // 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,
530 // 给计数器+1。
531 // 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,
532 // 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
533 // 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
534 // 话此时的limit=1, 也就是说当前只可以取0-6。
535 //
536 // 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。
537 int dfs(int pos, int pre, int status, int limit)
538 {
539     //已搜到尽头, 返回"是否找到了答案"这个状态。
540     if(pos < 1)
541         return status;
542
543     //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。
544     if(!limit && DP[pos][pre][status] != -1)
545         return DP[pos][pre][status];
546
547     int end = limit ? DIG[pos] : 9;
548     int ret = 0;
549
550     //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
551     //还有没有答案都无所谓了。而limit用&&是因为只有前面受限、当前受限才能
552     //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
553     //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。
554     //
555     //这里用"不要49"一题来做例子。
556     for(int i = 0; i <= end; i++)
557         ret += dfs(pos - 1, i, status || (pre == 4 && i == 9), limit && (i == end));
558
559     //DP里保存完整的、取到尽头的数据
560     if(!limit)
561         DP[pos][pre][status] = ret;
562
563     return ret;
564 }
565
566 //点分治
567 #include <cstdio>
568 #include <cstring>
569 #include <algorithm>
570 #include <vector>
571 using namespace std;
572 typedef long long ll;
573
574 struct Edge{
575     int to, next, len;
576 }edge[20500];
577
578 int head[20500];
579 bool vis[20500];
580 int siz[20500];
581 int f[20500];
582 int cnt, rt, sum, n, k;
583 ll ans;
584 ll vs[20500];

```



```

585     int vsc;
586     ll v[20500];
587     int vc;
588
589     void init() {
590         memset(head, -1, sizeof(head));
591         memset(vis, false, sizeof(vis));
592         cnt=0;
593         ans=0;
594     }
595
596     void add(int u, int v, int w) {
597         edge[cnt].to=v;
598         edge[cnt].len=w;
599         edge[cnt].next=head[u];
600         head[u]=cnt++;
601     }
602
603     void getrt(int u, int fa) {
604         siz[u]=1;
605         f[u]=0;
606         for(int i=head[u];~i;i=edge[i].next){
607             if(edge[i].to!=fa&&!vis[edge[i].to]){
608                 getrt(edge[i].to, u);
609                 siz[u]+=siz[edge[i].to];
610                 f[u]=max(f[u], siz[edge[i].to]);
611             }
612         }
613         f[u]=max(f[u], sum-siz[u]);
614         if(f[u]<f[rt])rt=u;
615     }
616
617     void getdeep(int u, int fa, ll len) {
618         vs[++vsc]=len;
619         v[++vc]=len;
620         for(int i=head[u];~i;i=edge[i].next){
621             if(edge[i].to!=fa&&!vis[edge[i].to]){
622                 getdeep(edge[i].to, u, len+edge[i].len);
623             }
624         }
625     }
626
627     void solve(int u) {
628         vis[u]=true;
629         vsc=0;
630         for(int i=head[u];~i;i=edge[i].next){
631             if(!vis[edge[i].to]){
632                 vc=0;
633                 getdeep(edge[i].to, u, edge[i].len);
634                 sort(v+1, v+1+vc);
635                 int l=1, r=vc;
636                 while(l<r){
637                     if(v[l]+v[r]<=k){
638                         ans+=(r-l);
639                         l++;
640                     }
641                     else r--;
642                 }
643             }
644         }
645         sort(vs+1, vs+1+vsc);
646         int l=1, r=vsc;
647         while(l<r){
648             if(vs[l]+vs[r]<=k){
649                 ans+=(r-l);
650                 l++;
651             }
652             else r--;
653         }
654         for(int i=1;i<=vsc;i++){
655             if(vs[i]<=k)ans++;
656         }
657         for(int i=head[u];~i;i=edge[i].next){

```

```

658         if(!vis[edge[i].to]){
659             rt=0, sum=siz[edge[i].to];
660             getrt(edge[i].to, 0);
661             solve(rt);
662         }
663     }
664 }
665
666 int main()
667 {
668     while(~scanf("%d%d", &n, &k)){
669         if(!n&&!k)break;
670         init();
671         for(int i=1;i<n;i++){
672             int u, v, w;
673             scanf("%d%d%d", &u, &v, &w);
674             add(u, v, w);
675             add(v, u, w);
676         }
677         f[0]=30000;
678         rt=0, sum=n;
679         getrt(1, 0);
680         solve(rt);
681         printf("%lld\n", ans);
682     }
683 }
684
685 //线性递推模板 $m^{2\log n}$ 
686 #include <bits/stdc++.h>
687 #pragma comment(linker, "/STACK:102400000,102400000")
688 using namespace std;
689 typedef long long ll;
690 #define LL long long
691 const int mod=1e9+7;
692 const int MOD=1e9+7;
693 const int MAXN=205;
694 ll n;
695 int u, d;
696 int a[100], b[100];
697 ll dp[205];
698 ll A[205];
699 ll C[205];
700
701 // given first m a[i] and coef (0 based)
702 // calc a[n] % MOD in  $O(m*m*\log(n))$ 
703 // a[n] = sum(c[m - i] * a[n - i]) i = 1....m
704 // a[m] = sum(c[i] * a[i]), i = 0.....m-1
705
706 LL linear_recurrence(LL n, int m, LL a[], LL c[], int p){ //n->a[i], m -> c[i]
707     LL v[MAXN] = {1 % MOD}, u[MAXN << 1], msk = !!n;
708     for(LL i = n; i > 1; i >>= 1) msk <<= 1;
709     for(LL x = 0; msk; msk >>= 1, x <<= 1){
710         fill_n(u, m << 1, 0);
711         int b = !(n & msk); x |= b;
712         if(x < m) u[x] = 1 % p;
713         else{
714             for(int i = 0; i < m; i++){
715                 for(int j = 0, t = i + b; j < m; ++j, ++t)
716                     u[t] = (u[t] + v[i] * v[j]) % MOD;
717             }
718             for(int i = (m << 1) - 1; i >= m; --i){
719                 for(int j = 0, t = i - m; j < m; ++j, ++t){
720                     u[t] = (u[t] + c[j] * u[i]) % MOD;
721                 }
722             }
723         }
724         copy(u, u+m, v);
725     }
726     LL ans = 0;
727     for(int i = 0; i < m; i++){
728         ans = (ans + v[i] * a[i]) % MOD;
729     }
730     return ans;

```

```

731 }
732
733 bool vis[205];
734 int main()
735 {
736     while(~scanf("%lld", &n)){
737         int ma=0;
738         memset(dp, 0, sizeof(dp));
739         memset(A, 0, sizeof(A));
740         memset(C, 0, sizeof(C));
741         memset(vis, false, sizeof(vis));
742         scanf("%d", &u);
743         for(int i=1;i<=u;i++){scanf("%d", &a[i]);}
744         scanf("%d", &d);
745         for(int i=1;i<=d;i++){scanf("%d", &b[i]);ma=max(b[i], ma);vis[b[i]]=true;}
746         dp[0]=1;
747         for(int i=1;i<=ma;i++){
748             for(int j=1;j<=u;j++){
749                 if(i<a[j])continue;
750                 dp[i]=(dp[i-a[j]]+dp[i])%mod;
751             }
752         }
753         vis[0]=true;
754         for(int i=0;i<=ma;i++)if(!vis[i])dp[i]=0;
755         for(int i=0;i<ma;i++){C[i]=dp[ma-i];}
756         A[0]=1;
757         for(int i=1;i<ma;i++){
758             for(int j=1;j<=i;j++){
759                 A[i]=(A[i]+A[i-j]*dp[j])%mod;
760             }
761         }
762         ll ans=linear_recurrence(n, ma, A, C, mod);
763         printf("%lld\n", ans);
764     }
765 }
766
767 //吉司机线段树
768 void update(int u, int ql, int qr, int c, int l, int r){
769     if(r<ql||qr<l||cut())return;
770     if(ql<=l&&r<=qr&&check()){
771         putlazy(u,c);
772         return;
773     }
774     int mid=(l+r)/2;
775     pushdown(u);
776     update(2*u, ql, qr, c, l, mid);
777     update(2*u+1, ql, qr, c, mid+1, r);
778     pushup(u);
779 }
780
781 //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：
782 //case 1: t大于最大值，此时区间不变；
783 //case
784 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
785 ;
786 //case 3: t大于严格次大值，小于最大值，这里可以打懒标记。
787 //
788 //考虑查询，只需维护最大值，最大值个数，严格次大值即可。
789 //吉司机宇宙线段树之王！
790 #include <bits/stdc++.h>
791 using namespace std;
792 const int maxn=1000005;
793 typedef long long ll;
794
795 int mx[maxn<<2];
796 int cnt[maxn<<2];
797 int se[maxn<<2];
798 int lazy[maxn<<2];
799 ll sum[maxn<<2];
800 int a[maxn];
801 int n, m;
802 void putlazy(int u, int t){

```

```

802     sum[u] -= 1LL * cnt[u] * (mx[u] - t);
803     mx[u] = t;
804     lazy[u] = t;
805 }
806
807 void pushdown(int u) {
808     if (lazy[u] == -1) return;
809     if (mx[2*u] > lazy[u]) {
810         sum[2*u] -= 1LL * cnt[2*u] * (mx[2*u] - lazy[u]);
811         mx[2*u] = lazy[u];
812         lazy[2*u] = lazy[u];
813     }
814     if (mx[2*u+1] > lazy[u]) {
815         sum[2*u+1] -= 1LL * cnt[2*u+1] * (mx[2*u+1] - lazy[u]);
816         mx[2*u+1] = lazy[u];
817         lazy[2*u+1] = lazy[u];
818     }
819 }
820 lazy[u] = -1;
821 }
822
823 void pushup(int u) {
824     if (mx[2*u] == mx[2*u+1]) {
825         mx[u] = mx[2*u];
826         cnt[u] = cnt[2*u] + cnt[2*u+1];
827         se[u] = max(se[2*u], se[2*u+1]);
828         sum[u] = sum[2*u] + sum[2*u+1];
829     }
830     else if (mx[2*u] > mx[2*u+1]) {
831         mx[u] = mx[2*u];
832         cnt[u] = cnt[2*u];
833         se[u] = max(se[2*u], mx[2*u+1]);
834         sum[u] = sum[2*u] + sum[2*u+1];
835     }
836     else {
837         mx[u] = mx[2*u+1];
838         cnt[u] = cnt[2*u+1];
839         se[u] = max(mx[2*u], se[2*u+1]);
840         sum[u] = sum[2*u] + sum[2*u+1];
841     }
842 }
843
844 void build(int u, int l, int r) {
845     lazy[u] = -1;
846     if (l == r) {
847         mx[u] = sum[u] = a[l];
848         cnt[u] = 1;
849         se[u] = -1;
850         return;
851     }
852     int mid = l + r >> 1;
853     build(2*u, l, mid);
854     build(2*u+1, mid+1, r);
855     pushup(u);
856 }
857
858 void update(int u, int ql, int qr, int t, int l, int r) {
859     if (ql > r || qr < l || mx[u] <= t) return;
860     if (ql <= l && r <= qr && se[u] < t) {
861         putlazy(u, t);
862         return;
863     }
864     pushdown(u);
865     int mid = l + r >> 1;
866     update(2*u, ql, qr, t, l, mid);
867     update(2*u+1, ql, qr, t, mid+1, r);
868     pushup(u);
869 }
870
871 int getmx(int u, int ql, int qr, int l, int r) {
872     if (ql > r || qr < l) return 0;
873     if (ql <= l && r <= qr) return mx[u];
874     pushdown(u);

```

```

875     int mid=l+r>>1;
876     int ans=0;
877     ans=max(ans, getmx(2*u, ql, qr, l, mid));
878     ans=max(ans, getmx(2*u+1, ql, qr, mid+1, r));
879     return ans;
880 }
881
882 ll getsum(int u, int ql, int qr, int l, int r){
883     if(ql>r||qr<l) return 0;
884     if(ql<=l&&r<=qr) return sum[u];
885     pushdown(u);
886     int mid=l+r>>1;
887     ll ans=0;
888     ans+=getsum(2*u, ql, qr, l, mid);
889     ans+=getsum(2*u+1, ql, qr, mid+1, r);
890     return ans;
891 }
892
893 int main(){
894     int T;
895     scanf("%d", &T);
896     while(T--){
897         scanf("%d%d", &n, &m);
898         for(int i=1;i<=n;i++) scanf("%d", &a[i]);
899         build(1, 1, n);
900         for(int i=1;i<=m;i++){
901             int tag;
902             scanf("%d", &tag);
903             if(tag==0){
904                 int x, y, t;
905                 scanf("%d%d%d", &x, &y, &t);
906                 update(1, x, y, t, 1, n);
907             }
908             else if(tag==1){
909                 int x, y;
910                 scanf("%d%d", &x, &y);
911                 printf("%d\n", getmx(1, x, y, 1, n));
912             }
913             else {
914                 int x, y;
915                 scanf("%d%d", &x, &y);
916                 printf("%lld\n", getsum(1, x, y, 1, n));
917             }
918         }
919     }
920 }
921
922 //树链剖分
923 //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
924 //input:
925 //1
926 //3
927 //1 2 1
928 //2 3 2
929 //QUERY 1 2
930 //CHANGE 1 3
931 //QUERY 1 2
932 //DONE
933 //
934 //Output:
935 //1
936 //3
937
938 #include <bits/stdc++.h>
939 using namespace std;
940 const int maxn=10500;
941
942 int dep[maxn],siz[maxn],fa[maxn],id[maxn],son[maxn],val[maxn],top[maxn];
943 int num,n;
944 int ma[maxn<<2];
945 vector<int>g[maxn];
946 struct edge{

```

```

947     int x,y,val;
948     void read() {
949         scanf("%d%d%d",&x,&y,&val);
950     }
951 }e[maxn];
952
953 void init() {
954     for(int i=1;i<=n;i++)g[i].clear();
955 }
956
957 void dfs1(int u,int f,int d){
958     dep[u]=d,siz[u]=1,son[u]=0,fa[u]=f;
959     for(int i=0;i<g[u].size();i++){
960         int v=g[u][i];
961         if(v==f)continue;
962         dfs1(v,u,d+1);
963         siz[u]+=siz[v];
964         if(siz[son[u]]<siz[v])son[u]=v;
965     }
966 }
967
968 void dfs2(int u,int tp){
969     top[u]=tp;
970     id[u]=++num;
971     if(son[u])dfs2(son[u],tp);
972     for(int i=0;i<g[u].size();i++){
973         int v=g[u][i];
974         if(v==fa[u]||v==son[u])continue;
975         dfs2(v,v);
976     }
977 }
978
979 void pushup(int u){
980     ma[u]=max(ma[2*u],ma[2*u+1]);
981 }
982
983 void build(int u,int l,int r){
984     if(l==r){
985         ma[u]=val[l];
986         return;
987     }
988     int mid=(l+r)/2;
989     build(2*u,l,mid);
990     build(2*u+1,mid+1,r);
991     pushup(u);
992 }
993
994 void update(int u,int p,int l,int r,int q){
995     if(p<l||p>r)return;
996     if(l==r){
997         ma[u]=q;
998         return;
999     }
1000     int mid=(l+r)/2;
1001     update(2*u,p,l,mid,q);
1002     update(2*u+1,p,mid+1,r,q);
1003     pushup(u);
1004 }
1005
1006 int query(int u,int ql,int qr,int l,int r){
1007     if(ql>r||qr<l)return 0;
1008     if(ql<=l&&r<=qr)return ma[u];
1009     int mid=(l+r)/2;
1010     return max(query(2*u,ql,qr,l,mid),query(2*u+1,ql,qr,mid+1,r));
1011 }
1012
1013
1014 int Query(int u,int v){
1015     int ret=0;
1016     int tp1=top[u],tp2=top[v];
1017     while(tp1!=tp2){
1018         if(dep[tp1]<dep[tp2]){
1019             swap(u,v);swap(tp1,tp2);

```

```

1020     }
1021     ret=max(ret,query(1,id[tp1],id[u],1,num));
1022     u=fa[tp1];
1023     tp1=top[u];
1024 }
1025 if(u==v) return ret;
1026 if(dep[u]<dep[v]) swap(u,v);
1027 ret=max(ret,query(1,id[son[v]],id[u],1,num));
1028 return ret;
1029 }
1030
1031
1032
1033
1034 int main() {
1035     int T;
1036     scanf("%d",&T);
1037     while(T--){
1038         scanf("%d",&n);
1039         init();
1040         for(int i=1;i<n;i++){
1041             e[i].read();
1042             g[e[i].x].push_back(e[i].y);
1043             g[e[i].y].push_back(e[i].x);
1044         }
1045         num=0;
1046         dfs1(1,0,1);
1047         dfs2(1,1);
1048         for(int i=1;i<n;i++){
1049             if(dep[e[i].x]<dep[e[i].y]) swap(e[i].x,e[i].y);
1050             val[id[e[i].x]]=e[i].val;
1051         }
1052         build(1,1,num);
1053         char tag[30];
1054         while(scanf("%s",tag)){
1055             if(tag[0]=='D') break;
1056             if(tag[0]=='Q'){
1057                 int x,y;
1058                 scanf("%d%d",&x,&y);
1059                 printf("%d\n",Query(x,y));
1060             }
1061             else {
1062                 int pos,x;
1063                 scanf("%d%d",&pos,&x);
1064                 update(1,id[e[pos].x],1,num,x);
1065             }
1066         }
1067     }
1068 }
1069
1070 //二分图匹配
1071 //用和为素数的二元组覆盖最多的数
1072 #include <bits/stdc++.h>
1073 using namespace std;
1074 const int maxn=3003;
1075 const int maxe=3003*3003;
1076 vector<int>g[maxn];
1077 int head[maxn], linker[maxn], vis[maxn], n, k, sum;
1078 void init(){
1079     memset(linker, -1, sizeof(linker));
1080     for(int i=0;i<n;i++) g[i].clear();
1081     sum=0;
1082 }
1083 void add(int u, int v){
1084     g[u].push_back(v);
1085 }
1086 int num[maxn];
1087 bool isprime[2000005];
1088 void init1(){
1089     memset(isprime, true, sizeof(isprime));
1090     isprime[0]=isprime[1]=false;
1091     for(int i=2;i<=2000000;i++){
1092         if(isprime[i]){

```

```

1093         for(int j=2;j*i<=2000000;j++) isprime[i*j]=false;
1094     }
1095 }
1096 }
1097 bool cmp(int x, int y){
1098     return x>y;
1099 }
1100 int Find(int u){
1101     for(int i=0;i<g[u].size();i++){
1102         int v=g[u][i];
1103         if(!vis[v]){
1104             vis[v]=1;
1105             if(linker[v]==-1||Find(linker[v])){
1106                 linker[v]=u;
1107                 linker[u]=v;
1108                 return 1;
1109             }
1110         }
1111     }
1112     return 0;
1113 }
1114 int match(){
1115     int ans=0;
1116     for(int i=0;i<n;i++){
1117         if(num[i]&1&&linker[i]==-1){
1118             memset(vis, 0, sizeof(vis));
1119             vis[i]=true;
1120             ans+=Find(i);
1121         }
1122     }
1123     return ans;
1124 }
1125 int main(){
1126     init1();
1127     int T;
1128     scanf("%d", &T);
1129     while(T--){
1130         scanf("%d%d", &n, &k);
1131         init();
1132         for(int i=0;i<n;i++){
1133             scanf("%d", &num[i]);
1134         }
1135         sort(num, num+n, cmp);
1136         for(int i=0;i<n;i++){
1137             bool ok=false;
1138             for(int j=0;j<n;j++){
1139                 if(i==j) continue;
1140                 if(isprime[num[i]+num[j]]){
1141                     ok=true;
1142                     add(i, j);
1143                 }
1144             }
1145             if(ok) sum++;
1146         }
1147         for(int i=0;i<n;i++) sort(g[i].begin(), g[i].end());
1148         if(sum<=k){
1149             printf("%d\n", sum);
1150         }
1151         else {
1152             int m=match();
1153             if(m>=k) printf("%d\n", 2*k);
1154             else if(sum-m<=k) printf("%d\n", sum);
1155             else printf("%d\n", k+m);
1156         }
1157     }
1158 }
1159
1160 //bzoj1010
1161 //决策单调性
1162 //
1163 //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

$$x = j - i + \sum_{i \leq k \leq j} C_k$$

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。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。

```
1164 //
1165 //input
1166 //5 4
1167 //3
1168 //4
1169 //2
1170 //1
1171 //4
1172 //
1173 //output
1174 //1
1175 //
1176 #include <bits/stdc++.h>
1177 using namespace std;
1178 typedef long long ll;
1179 const int maxn=50005;
1180 ll a[maxn], s[maxn], dp[maxn], l;
1181 int n;
1182 struct node{
1183     int l, r, p;
1184 }q[maxn];
1185 ll cal(int j, int i){
1186     return dp[j]+(s[i]-s[j]+i-j-1-l)*(s[i]-s[j]+i-j-1-l);
1187 }
1188 int bisearch(node a, int i){
1189     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
1190     while(l<=r){
1191         if(r-l<=1){
1192             if(cal(i, l)<cal(a.p, l))tag=l;
1193             else tag=r;
1194             break;
1195         }
1196         int mid=l+r>>1;
1197         if(cal(i, mid)<cal(a.p, mid))r=mid;
1198         else l=mid;
1199     }
1200     return tag;
1201 }
1202 void solve(){
1203     int head=1, tail=0;
1204     q[++tail]=(node){0, n, 0};
1205     for(int i=1;i<=n;i++){
1206         if(i>q[head].r)head++;
1207         dp[i]=cal(q[head].p, i);
1208         /*for(int i=head;i<=tail;i++){
1209             printf("node %d %d %d\n", q[i].l, q[i].r, q[i].p);
1210         }
1211         cout<<i<<' '<<q[head].p<<' '<<dp[i]<<endl;*/
1212         if(head>tail||cal(i, n)<cal(q[tail].p, n)){
1213             while(head<=tail&&cal(i, q[tail].l)<cal(q[tail].p, q[tail].l))tail--;
1214             if(head<=tail){
1215                 int t=bisearch(q[tail], i);
1216                 q[tail].r=t-1;
1217                 q[++tail]=(node){t, n, i};
1218             }
1219             else q[++tail]=(node){i, n, i};
1220         }
1221     }
1222 }
1223 int main(){
1224     scanf("%d%lld", &n, &l);
1225     for(ll i=1;i<=n;i++){
1226         scanf("%lld", &a[i]);
1227         s[i]=s[i-1]+a[i];
1228     }
1229     solve();
```

```

1230     printf("%lld\n", dp[n]);
1231 }
1232
1233
1234
1235 // #include <bitset>
1236 // using std::bitset;
1237 // bitset<n>b;
1238 // b.any() any 1?
1239 // b.none() no 1?
1240 // b.count() number of 1?
1241 // b.size() size?
1242 // b.set() set all pos 1
1243 // b.reset() set all pos 0
1244 // b.flip() 1->0, 0->1
1245
1246 // k中浓度为ai/1000的液体配成n/1000的浓度至少需要多少升
1247 // 每种溶液均为整数升
1248 // 易知若有解必小于1000, dp过程用bitset加速
1249
1250 #include <bits/stdc++.h>
1251 using namespace std;
1252
1253 int n, k;
1254 bool a[1050];
1255 bitset<2050>b[2];
1256
1257 int main() {
1258     scanf("%d%d", &n, &k);
1259     for(int i=1; i<=k; i++) {
1260         int x;
1261         scanf("%d", &x);
1262         a[x]=true;
1263     }
1264     b[0][1000]=1;
1265     for(int i=0; i<=1000; i++) {
1266         if(i!=0 && b[i%2][1000]) {
1267             printf("%d\n", i);
1268             return 0;
1269         }
1270         int now=i%2;
1271         b[now^1].reset();
1272         for(int j=0; j<=1000; j++) {
1273             if(a[j]) {
1274                 b[now^1] |= (b[now] << j) >> n;
1275             }
1276         }
1277     }
1278     printf("-1\n");
1279 }
1280
1281
1282
1283 // 好题
1284 // 问题归结为:
1285 // 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
1286 // 用多重背包的按位拆分+bitset可以达到 $O(k \cdot \sigma(\log A_i) / 64)$ 
1287 // 而由于 $\sigma(i \cdot A_i) = n$ , 最坏情况 $O(\sigma(\log A_i)) \sim O(\sqrt{n})$ 
1288 // 所以总复杂度是 $O(k \cdot \sqrt{n} / 64)$ 
1289 #include <bits/stdc++.h>
1290 using namespace std;
1291
1292 int a[1050000];
1293 bool vis[1050000];
1294 int cnt[1050000];
1295 bitset<1050000>b;
1296 int n, k;
1297 int ma=0, mi, bonus=0;
1298
1299 int main() {
1300     scanf("%d%d", &n, &k);
1301     for(int i=1; i<=n; i++) scanf("%d", &a[i]);
1302     for(int i=1; i<=n; i++) {

```

```

1303         if(vis[i])continue;
1304         int tmp=i;
1305         int len=0;
1306         while(!vis[tmp]){
1307             vis[tmp]=true;
1308             len++;
1309             tmp=a[tmp];
1310         }
1311         cnt[len]++;
1312         bonus+=len/2;
1313     }
1314     ma=min(n, k+min(bonus, k));
1315     b[0]=1;
1316     for(int i=1;i<=n;i++){
1317         if(!cnt[i])continue;
1318         int k=1;
1319         while(k<cnt[i]){
1320             b|=b<<k*i;
1321             cnt[i]-=k;
1322             k<<=1;
1323         }
1324         b|=b<<cnt[i]*i;
1325     }
1326     if(b[k])mi=k;
1327     else mi=k+1;
1328     printf("%d %d\n", mi, ma);
1329 }
1330
1331 //树状数组区间修改区间查询
1332 //sum[n]=sigma(a[i])+sigma((n+1-i)d[i]);
1333 //其中a[i]是原数组元素
1334 //d[i]是[i, n]的共同增量
1335 #include <bits/stdc++.h>
1336 using namespace std;
1337 typedef long long ll;
1338 ll a[205000];
1339 ll d[205000];
1340 ll di[205000];
1341 int n, q;
1342
1343 int lowbit(int x){
1344     return x&(-x);
1345 }
1346
1347 void update(int x, int num){
1348     ll add=1LL*num;
1349     ll addi=1LL*num*x;
1350     while(x<=n){
1351         d[x]+=add;
1352         di[x]+=addi;
1353         x+=lowbit(x);
1354     }
1355 }
1356
1357 ll query(int x){
1358     ll ans=a[x];
1359     ll tmp=x;
1360     while(tmp){
1361         ans+=d[tmp]*(x+1);
1362         ans+=di[tmp]*(-1);
1363         tmp-=lowbit(tmp);
1364     }
1365     return ans;
1366 }
1367
1368 int main()
1369 {
1370     scanf("%d", &n);
1371     for(int i=1;i<=n;i++){
1372         scanf("%lld", &a[i]);
1373         a[i]+=a[i-1];
1374     }
1375     scanf("%d", &q);

```

```

1376     while(q--){
1377         int tag;
1378         scanf("%d", &tag);
1379         if(tag==1){
1380             int x, y, num;
1381             scanf("%d%d%d", &x, &y, &num);
1382             update(x, num);
1383             update(y+1, -num);
1384         }
1385         else {
1386             int x, y;
1387             scanf("%d%d", &x, &y);
1388             printf("%lld\n", query(y)-query(x-1));
1389         }
1390     }
1391 }
1392
1393 //SA
1394 #include <cstdio>
1395 #include <cstring>
1396 #include <algorithm>
1397 using namespace std;
1398 const int maxn=20500;
1399 int s[maxn];
1400 int sa[maxn], t[maxn], t2[maxn], c[maxn], n;
1401 int Rank[maxn], height[maxn];
1402 void build_sa(int m, int n){
1403     int i, *x=t, *y=t2;
1404     for(i=0;i<m;i++) c[i]=0;
1405     for(i=0;i<n;i++) c[x[i]=s[i]]++;
1406     for(i=1;i<m;i++) c[i]+=c[i-1];
1407     for(i=n-1;i>=0;i--) sa[--c[x[i]]]=i;
1408     for(int k=1;k<=n;k<=<=1){
1409         int p=0;
1410         for(i=n-k;i<n;i++) y[p++]=i;
1411         for(i=0;i<n;i++) if(sa[i]>=k) y[p++]=sa[i]-k;
1412         for(i=0;i<m;i++) c[i]=0;
1413         for(i=0;i<n;i++) c[x[y[i]]]++;
1414         for(i=1;i<m;i++) c[i]+=c[i-1];
1415         for(i=n-1;i>=0;i--) sa[--c[x[y[i]]]]=y[i];
1416         swap(x, y);
1417         p=1; x[sa[0]]=0;
1418         for(i=1;i<n;i++)
1419             x[sa[i]]=y[sa[i-1]]==y[sa[i]]&&y[sa[i-1]+k]==y[sa[i]+k]?p-1:p++;
1420         if(p>=n) break;
1421         m=p;
1422     }
1423 }
1424 void geth(int n){
1425     int i, j, k=0;
1426     for(i=0;i<n;i++) Rank[sa[i]]=i;
1427     for(i=0;i<n;i++){
1428         if(k) k--;
1429         if(Rank[i]==0) break;
1430         j=sa[Rank[i]-1];
1431         while(s[i+k]==s[j+k]) k++;
1432         height[Rank[i]]=k;
1433     }
1434 }
1435
1436 int main()
1437 {
1438     n=8;
1439     s[8]=0;
1440     s[0]=s[1]=2;
1441     s[2]=s[3]=s[4]=s[5]=1;
1442     s[6]=s[7]=3;
1443     build_sa(200, n+1);
1444     geth(n+1);
1445     for(int i=0;i<=n;i++){
1446         printf("%d ", sa[i]);
1447     }
1448     printf("\n");

```

```
1449     for(int i=1;i<=n;i++){
1450         printf("%d ", height[i]);
1451     }
1452     printf("\n");
1453
1454 }
1455
1456
1457
```