

```

1 //输入外挂
2 //扩展欧几里得
3 //素数线性筛
4 //中国剩余定理
5 //逆元
6 //FFT
7 //NTT
8 //MILLER RABIN
9 //AC自动机
10 //数位DP
11 //点分治
12 //线性递推模板 $m^{2\log n}$ 
13 //吉司机线段树
14 //树链剖分
15 //二分图匹配
16 //决策单调性
17 //bitset
18 //树状数组区间修改区间查询
19 //SA
20
21 //输入外挂
22 #include <bits/stdc++.h>
23 using namespace std;
24 template <class T>
25 inline bool scan_d(T &ret){
26     char c; int sgn;
27     if(c=getchar(), c==EOF)return 0;
28     while(c!='-' && (c<'0' || c>'9'))c=getchar();
29     sgn=(c=='-')?-1:1;
30     ret=(c=='-')?0:(c-'0');
31     while(c=getchar(), c>='0' && c<='9')ret=ret*10+c-'0';
32     ret*=sgn;
33     return 1;
34 }
35
36 int main(){
37     int a, b;
38     while(1){
39         scan_d(a);
40         scan_d(b);
41         cout<<a+b<<endl;
42     }
43 }
44
45
46 //扩展欧几里得
47 //求 $ax+by=\gcd(a, b)$ 的解
48 void exgcd(ll a, ll b, ll &x, ll &y){
49     if(b==0){
50         x=1, y=0;
51         return ;
52     }
53     exgcd(b, a%b, x, y);
54     ll tmp=x;
55     x=y;
56     y=tmp-(a/b)*y;
57 }
58
59 //素数线性筛
60 const int maxn=205000;
61
62 bool vis[maxn];
63 int prime[maxn];
64 int tot;
65 void init(){
66     memset(vis, false, sizeof(vis));
67     tot=0;
68     for(int i=2; i<maxn; i++){
69         if(!vis[i])prime[++tot]=i;
70         for(int j=1; j<=tot && prime[j]*i<maxn; j++){
71             vis[prime[j]*i]=true;
72             if(i%prime[j]==0)break;
73         }
74     }
75 }

```

```

74     }
75 }
76
77 //中国剩余定理
78 //正整数m1, m2, ..., mk两两互素, 则同余方程组
79 //
80 //x%m1==a1
81 //x%m2==a2
82 //
83 //x%mk==ak
84 //
85 //在模M=m1*m1*...*mk下的解唯一
86 //
87 //x%M==sigma(ai*Mi*inv(Mi))
88 //
89 //Mi=M/mi, inv(Mi)为Mi模mi的逆元
90 int CRT(int a[], int m[], int n){
91     int M=1;
92     int ans=0;
93     for(int i=1;i<=n;i++) M*=m[i];
94     for(int i=1;i<=n;i++){
95         int x, y;
96         int Mi=M/m[i];
97         exgcd(Mi, m[i], x, y); //m[i]不一定是素数, 故要用exgcd求逆元
98         ans=(ans+a[i]*Mi*x)%M;
99     }
100     if(ans<0) ans+=M;
101     return ans;
102 }
103
104 //卢卡斯定理
105 //rm to do
106
107 //逆元
108 //ans=(a/b)%m=(a%(mb))/b
109 //fermat小定理
110 //exgcd
111 //o(n) 预处理mod的逆元
112 int mod;
113 int inv[maxn];
114
115 void init(){
116     inv[1] = 1;
117     for(int i=2;i<mod;i++){
118         inv[i]=inv[mod%i]*(mod-mod/i)%mod;
119     }
120 }
121
122 //欧拉函数
123 //rm to do
124
125 //莫比乌斯函数
126 //rm to do
127
128 //威佐夫博弈
129 //rm to do
130
131 //反NIM博弈
132 //rm to do
133
134 //FFT
135 #include <stdio.h>
136 #include <string.h>
137 #include <iostream>
138 #include <algorithm>
139 #include <math.h>
140 using namespace std;
141
142 const double PI = acos(-1.0);
143 //复数结构体
144 struct complex
145 {
146     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 {

```

```

220 while (scanf("%s%s",str1,str2)==2)
221 {
222     int len1 = strlen(str1);
223     int len2 = strlen(str2);
224     int len = 1;
225     while(len < len1*2 || len < len2*2) len<<=1;
226     for(int i = 0;i < len1;i++)
227         x1[i] = complex(str1[len1-1-i]-'0',0);
228     for(int i = len1;i < len;i++)
229         x1[i] = complex(0,0);
230     for(int i = 0;i < len2;i++)
231         x2[i] = complex(str2[len2-1-i]-'0',0);
232     for(int i = len2;i < len;i++)
233         x2[i] = complex(0,0);
234     //求DFT
235     fft(x1,len,1);
236     fft(x2,len,1);
237     for(int i = 0;i < len;i++)
238         x1[i] = x1[i]*x2[i];
239     fft(x1,len,-1);
240     for(int i = 0;i < len;i++)
241         sum[i] = (int)(x1[i].r+0.5);
242     for(int i = 0;i < len;i++)
243     {
244         sum[i+1]+=sum[i]/10;
245         sum[i]%=10;
246     }
247     len = len1+len2-1;
248     while(sum[len] <= 0 && len > 0) len--;
249     for(int i = len;i >= 0;i--)
250         printf("%c",sum[i]+'0');
251     printf("\n");
252 }
253 return 0;
254 }
255
256 //NTT
257 #include<cstdio>
258 #include<cstring>
259 #include<algorithm>
260 using namespace std;
261 #define MAXN 262144
262
263 const long long P=50000000001507329LL; // 190734863287 * 2 ^ 18 + 1
264 //const int P=1004535809; // 479 * 2 ^ 21 + 1
265 //const int P=998244353; // 119 * 2 ^ 23 + 1
266 const int G=3;
267
268 long long mul(long long x,long long y)
269 {
270     return (x*y-(long long)(x/(long double)P*y+1e-3)*P+P)%P;
271 }
272 long long qpow(long long x,long long k,long long p)
273 {
274     long long ret=1;
275     while(k)
276     {
277         if(k&1) ret=mul(ret,x);
278         k>>=1;
279         x=mul(x,x);
280     }
281     return ret;
282 }
283
284 long long wn[25];
285 void getwn()
286 {
287     for(int i=1; i<=18; ++i)
288     {
289         int t=1<<i;
290         wn[i]=qpow(G,(P-1)/t,P);
291     }
292 }

```

```

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();

```

```

366     debug();
367 }
368
369 //FWT
370 //rm to do
371
372 //Miller Rabin
373 //Pollard_rho
374
375 #include <bits/stdc++.h>
376 using namespace std;
377
378 typedef unsigned long long ll;
379
380 const int T=10;
381
382 ll mul(ll x,ll y, ll P)
383 {
384     return (x*y-(ll) (x/(ll) P*y+1e-3)*P+P)%P;
385 }
386
387 ll exp(ll x, ll y, ll mod){
388     ll ans=1;
389     ll base=x;
390     while(y){
391         if(y&1){
392             ans=mul(ans, base, mod);
393         }
394         y>>=1;
395         base=mul(base, base, mod);
396     }
397     return ans;
398 }
399
400 bool miller_rabin(ll n)
401 {
402     if(n==2)return true;
403     if(n<2||!(n&1))return false;
404     ll m=n-1;
405     ll k=0;
406     while(!(m&1)){
407         k++;
408         m>>=1;
409     }
410     for(int i=0;i<T;i++){
411         ll a=rand()%(n-1)+1;
412         ll x=exp(a, m, n);
413         ll y=0;
414         for(int j=1;j<=k;j++){
415             y=mul(x, x, n);
416             if(y==1&&x!=1&&x!=n-1)return false;
417             x=y;
418         }
419         if(y!=1)return false;
420     }
421     return true;
422 }
423
424 ll Pollard_rho(ll x,ll c){
425     ll i=1,k=2;
426     ll x0=rand()%x;
427     ll y=x0;
428     while(1){
429         i++;
430         x0=(mul(x0,x0,x)+c)%x;
431         ll d=__gcd(y-x0,x);
432         if(d!=1&&d!=x) return d;
433         if(y==x0) return x;
434         if(i==k){y=x0;k+=k;}
435     }
436 }
437
438 void debug(){

```

```

439     ll n;
440     while(cin>>n){
441         //if(miller_rabin(n)) cout<<"YES\n";
442         //else cout<<"NO\n";
443         cout<<Pollard_rho(n, rand()%(n-1)+1)<<endl;
444     }
445 }
446
447 int main(){
448     srand(time(0));
449     debug();
450 }
451
452
453
454 //AC自动机
455 //查询模式串出现次数
456 #include <bits/stdc++.h>
457 using namespace std;
458 struct Trie{
459     int ch[50500][27];
460     int fail[50500];
461     int end[50500];
462     int sz;
463     int newnode(){
464         for(int i=0;i<27;i++) ch[sz][i]=-1;
465         end[sz]=-1;
466         return sz++;
467     }
468     void init(){
469         sz=0;
470         newnode();
471     }
472     int idx(int c){
473         if(c>='A'&&c<='Z') return c-'A';
474         return 26;
475     }
476     void insert(char s[], int id){
477         int len=strlen(s);
478         int u=0;
479         for(int i=0;i<len;i++){
480             int c=idx(s[i]);
481             if(ch[u][c]==-1){
482                 ch[u][c]=newnode();
483             }
484             u=ch[u][c];
485         }
486         end[u]=id;
487     }
488     void build(){
489         queue<int>q;
490         fail[0]=0;
491         for(int i=0;i<27;i++){
492             if(ch[0][i]==-1) ch[0][i]=0;
493             else {
494                 fail[ch[0][i]]=0;
495                 q.push(ch[0][i]);
496             }
497         }
498         while(!q.empty()){
499             int u=q.front();
500             q.pop();
501             for(int i=0;i<27;i++){
502                 if(ch[u][i]==-1){
503                     ch[u][i]=ch[fail[u]][i];
504                 }
505                 else {
506                     fail[ch[u][i]]=ch[fail[u]][i];
507                     q.push(ch[u][i]);
508                 }
509             }
510         }
511     }

```

```

512 };
513
514 Trie T;
515 char s[1050][55];
516 char str[2050000];
517 int cnt[1050];
518 int n;
519
520 int main()
521 {
522     while(~scanf("%d", &n)){
523         memset(cnt, 0, sizeof(cnt));
524         T.init();
525         for(int i=1;i<=n;i++){
526             scanf("%s", s[i]);
527             T.insert(s[i], i);
528         }
529         T.build();
530         scanf("%s", str);
531         int len=strlen(str);
532         int u=0;
533         for(int i=0;i<len;i++){
534             int c=T.idx(str[i]);
535             u=T.ch[u][c];
536             int tmp=u;
537             while(tmp){
538                 if(T.end[tmp]!=-1){
539                     cnt[T.end[tmp]]++;
540                 }
541                 tmp=T.fail[tmp];
542             }
543         }
544         for(int i=1;i<=n;i++){
545             if(cnt[i]){
546                 printf("%s: %d\n", s[i], cnt[i]);
547             }
548         }
549     }
550 }
551
552 //数位DP
553 // pos = 当前处理的位置(一般从高位到低位)
554 // pre = 上一个位的数字(更高的那一位)
555 // 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,
556 // 给计数器+1。
557 // 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,
558 // 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
559 // 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
560 // 话此时的limit=1,也就是说当前只可以取0-6。
561 //
562 // 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。
563 int dfs(int pos,int pre,int status,int limit)
564 {
565     //已搜到尽头,返回"是否找到了答案"这个状态。
566     if(pos < 1)
567         return status;
568
569     //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。
570     if(!limit && DP[pos][pre][status] != -1)
571         return DP[pos][pre][status];
572
573     int end = limit ? DIG[pos] : 9;
574     int ret = 0;
575
576     //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
577     //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
578     //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
579     //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。
580     //
581     //这里用"不要49"一题来做例子。
582     for(int i = 0; i <= end; i ++){
583         ret += dfs(pos - 1,i,status || (pre == 4 && i == 9),limit && (i == end));
584     }

```



```

585 //DP里保存完整的、取到尽头的数据
586 if(!limit)
587     DP[pos][pre][status] = ret;
588
589     return    ret;
590 }
591
592 //点分治
593 #include <cstdio>
594 #include <cstring>
595 #include <algorithm>
596 #include <vector>
597 using namespace std;
598 typedef long long ll;
599
600 struct Edge{
601     int to, next, len;
602 }edge[20500];
603
604 int head[20500];
605 bool vis[20500];
606 int siz[20500];
607 int f[20500];
608 int cnt, rt, sum, n, k;
609 ll ans;
610 ll vs[20500];
611 int vsc;
612 ll v[20500];
613 int vc;
614
615 void init(){
616     memset(head, -1, sizeof(head));
617     memset(vis, false, sizeof(vis));
618     cnt=0;
619     ans=0;
620 }
621
622 void add(int u, int v, int w){
623     edge[cnt].to=v;
624     edge[cnt].len=w;
625     edge[cnt].next=head[u];
626     head[u]=cnt++;
627 }
628
629 void getrt(int u, int fa){
630     siz[u]=1;
631     f[u]=0;
632     for(int i=head[u];~i;i=edge[i].next){
633         if(edge[i].to!=fa&&!vis[edge[i].to]){
634             getrt(edge[i].to, u);
635             siz[u]+=siz[edge[i].to];
636             f[u]=max(f[u], siz[edge[i].to]);
637         }
638     }
639     f[u]=max(f[u], sum-siz[u]);
640     if(f[u]<f[rt])rt=u;
641 }
642
643 void getdeep(int u, int fa, ll len){
644     vs[++vsc]=len;
645     v[++vc]=len;
646     for(int i=head[u];~i;i=edge[i].next){
647         if(edge[i].to!=fa&&!vis[edge[i].to]){
648             getdeep(edge[i].to, u, len+edge[i].len);
649         }
650     }
651 }
652
653 void solve(int u){
654     vis[u]=true;
655     vsc=0;
656     for(int i=head[u];~i;i=edge[i].next){
657         if(!vis[edge[i].to]){

```

```

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

```

```

731
732 LL linear_recurrence(LL n, int m, LL a[], LL c[], int p){ //n->a[i], m -> c[i]
733     LL v[MAXN] = {1 % MOD}, u[MAXN << 1], msk = !!n;
734     for(LL i = n; i > 1; i >>= 1) msk <<= 1;
735     for(LL x = 0; msk; msk >>= 1, x <<= 1){
736         fill_n(u, m << 1, 0);
737         int b = !(n & msk); x |= b;
738         if(x < m) u[x] = 1 % p;
739         else{
740             for(int i = 0; i < m; i++){
741                 for(int j = 0, t = i + b; j < m; ++j, ++t)
742                     u[t] = (u[t] + v[i] * v[j]) % MOD;
743             }
744             for(int i = (m << 1) - 1; i >= m; --i){
745                 for(int j = 0, t = i - m; j < m; ++j, ++t){
746                     u[t] = (u[t] + c[j] * u[i]) % MOD;
747                 }
748             }
749         }
750         copy(u, u+m, v);
751     }
752     LL ans = 0;
753     for(int i = 0; i < m; i++){
754         ans = (ans + v[i] * a[i]) % MOD;
755     }
756     return ans;
757 }
758
759 bool vis[205];
760 int main()
761 {
762     while(~scanf("%lld", &n)){
763         int ma=0;
764         memset(dp, 0, sizeof(dp));
765         memset(A, 0, sizeof(A));
766         memset(C, 0, sizeof(C));
767         memset(vis, false, sizeof(vis));
768         scanf("%d", &u);
769         for(int i=1;i<=u;i++){scanf("%d", &a[i]);}
770         scanf("%d", &d);
771         for(int i=1;i<=d;i++){scanf("%d", &b[i]);ma=max(b[i], ma);vis[b[i]]=true;}
772         dp[0]=1;
773         for(int i=1;i<=ma;i++){
774             for(int j=1;j<=u;j++){
775                 if(i<a[j])continue;
776                 dp[i]=(dp[i-a[j]]+dp[i])%mod;
777             }
778         }
779         vis[0]=true;
780         for(int i=0;i<=ma;i++)if(!vis[i])dp[i]=0;
781         for(int i=0;i<=ma;i++){C[i]=dp[ma-i];}
782         A[0]=1;
783         for(int i=1;i<=ma;i++){
784             for(int j=1;j<=i;j++){
785                 A[i]=(A[i]+A[i-j]*dp[j])%mod;
786             }
787         }
788         ll ans=linear_recurrence(n, ma, A, C, mod);
789         printf("%lld\n", ans);
790     }
791 }
792
793 //吉司机线段树
794 void update(int u, int ql, int qr, int c, int l, int r){
795     if(r<ql||qr<l||cut())return;
796     if(ql<=l&&r<=qr&&check()){
797         putlazy(u,c);
798         return;
799     }
800     int mid=(l+r)/2;
801     pushdown(u);
802     update(2*u, ql, qr, c, l, mid);
803     update(2*u+1, ql, qr, c, mid+1, r);

```

```

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

```

```

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

```

```

948 //树链剖分
949 //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
950 //input:
951 //1
952 //3
953 //1 2 1
954 //2 3 2
955 //QUERY 1 2
956 //CHANGE 1 3
957 //QUERY 1 2
958 //DONE
959 //
960 //Output:
961 //1
962 //3
963
964 #include <bits/stdc++.h>
965 using namespace std;
966 const int maxn=10500;
967
968 int dep[maxn],siz[maxn],fa[maxn],id[maxn],son[maxn],val[maxn],top[maxn];
969 int num,n;
970 int ma[maxn<<2];
971 vector<int>g[maxn];
972 struct edge{
973     int x,y,val;
974     void read(){
975         scanf("%d%d%d",&x,&y,&val);
976     }
977 }e[maxn];
978
979 void init(){
980     for(int i=1;i<=n;i++)g[i].clear();
981 }
982
983 void dfs1(int u,int f,int d){
984     dep[u]=d,siz[u]=1,son[u]=0,fa[u]=f;
985     for(int i=0;i<g[u].size();i++){
986         int v=g[u][i];
987         if(v==f)continue;
988         dfs1(v,u,d+1);
989         siz[u]+=siz[v];
990         if(siz[son[u]]<siz[v])son[u]=v;
991     }
992 }
993
994 void dfs2(int u,int tp){
995     top[u]=tp;
996     id[u]=++num;
997     if(son[u])dfs2(son[u],tp);
998     for(int i=0;i<g[u].size();i++){
999         int v=g[u][i];
1000         if(v==fa[u]||v==son[u])continue;
1001         dfs2(v,v);
1002     }
1003 }
1004
1005 void pushup(int u){
1006     ma[u]=max(ma[2*u],ma[2*u+1]);
1007 }
1008
1009 void build(int u,int l,int r){
1010     if(l==r){
1011         ma[u]=val[l];
1012         return ;
1013     }
1014     int mid=(l+r)/2;
1015     build(2*u,l,mid);
1016     build(2*u+1,mid+1,r);
1017     pushup(u);
1018 }
1019

```

```

1020 void update(int u,int p,int l,int r,int q){
1021     if(p<l||p>r) return;
1022     if(l==r){
1023         ma[u]=q;
1024         return;
1025     }
1026     int mid=(l+r)/2;
1027     update(2*u,p,l,mid,q);
1028     update(2*u+1,p,mid+1,r,q);
1029     pushup(u);
1030 }
1031
1032 int query(int u,int ql,int qr,int l,int r){
1033     if(ql>r||qr<l) return 0;
1034     if(ql<=l&&r<=qr) return ma[u];
1035     int mid=(l+r)/2;
1036     return max(query(2*u,ql,qr,l,mid),query(2*u+1,ql,qr,mid+1,r));
1037 }
1038
1039
1040 int Query(int u,int v){
1041     int ret=0;
1042     int tp1=top[u],tp2=top[v];
1043     while(tp1!=tp2){
1044         if(dep[tp1]<dep[tp2]){
1045             swap(u,v);swap(tp1,tp2);
1046         }
1047         ret=max(ret,query(1,id[tp1],id[u],1,num));
1048         u=fa[tp1];
1049         tp1=top[u];
1050     }
1051     if(u==v) return ret;
1052     if(dep[u]<dep[v]) swap(u,v);
1053     ret=max(ret,query(1,id[son[v]],id[u],1,num));
1054     return ret;
1055 }
1056
1057
1058
1059
1060 int main(){
1061     int T;
1062     scanf("%d",&T);
1063     while(T--){
1064         scanf("%d",&n);
1065         init();
1066         for(int i=1;i<n;i++){
1067             e[i].read();
1068             g[e[i].x].push_back(e[i].y);
1069             g[e[i].y].push_back(e[i].x);
1070         }
1071         num=0;
1072         dfs1(1,0,1);
1073         dfs2(1,1);
1074         for(int i=1;i<n;i++){
1075             if(dep[e[i].x]<dep[e[i].y]) swap(e[i].x,e[i].y);
1076             val[id[e[i].x]]=e[i].val;
1077         }
1078         build(1,1,num);
1079         char tag[30];
1080         while(scanf("%s",tag)){
1081             if(tag[0]=='D') break;
1082             if(tag[0]=='Q'){
1083                 int x,y;
1084                 scanf("%d%d",&x,&y);
1085                 printf("%d\n",Query(x,y));
1086             }
1087             else {
1088                 int pos,x;
1089                 scanf("%d%d",&pos,&x);
1090                 update(1,id[e[pos].x],1,num,x);
1091             }
1092         }

```

```

1093     }
1094 }
1095
1096 //二分图匹配
1097 //用和为素数的二元组覆盖最多的数
1098 #include <bits/stdc++.h>
1099 using namespace std;
1100 const int maxn=3003;
1101 const int maxe=3003*3003;
1102 vector<int>g[maxn];
1103 int head[maxn], linker[maxn], vis[maxn], n, k, sum;
1104 void init(){
1105     memset(linker, -1, sizeof(linker));
1106     for(int i=0;i<n;i++)g[i].clear();
1107     sum=0;
1108 }
1109 void add(int u, int v){
1110     g[u].push_back(v);
1111 }
1112 int num[maxn];
1113 bool isprime[2000005];
1114 void init1(){
1115     memset(isprime, true, sizeof(isprime));
1116     isprime[0]=isprime[1]=false;
1117     for(int i=2;i<=2000000;i++){
1118         if(isprime[i]){
1119             for(int j=2;j*i<=2000000;j++)isprime[i*j]=false;
1120         }
1121     }
1122 }
1123 bool cmp(int x, int y){
1124     return x>y;
1125 }
1126 int Find(int u){
1127     for(int i=0;i<g[u].size();i++){
1128         int v=g[u][i];
1129         if(!vis[v]){
1130             vis[v]=1;
1131             if(linker[v]==-1||Find(linker[v])){
1132                 linker[v]=u;
1133                 linker[u]=v;
1134                 return 1;
1135             }
1136         }
1137     }
1138     return 0;
1139 }
1140 int match(){
1141     int ans=0;
1142     for(int i=0;i<n;i++){
1143         if(num[i]&1&&linker[i]==-1){
1144             memset(vis, 0, sizeof(vis));
1145             vis[i]=true;
1146             ans+=Find(i);
1147         }
1148     }
1149     return ans;
1150 }
1151 int main(){
1152     init1();
1153     int T;
1154     scanf("%d", &T);
1155     while(T--){
1156         scanf("%d%d", &n, &k);
1157         init();
1158         for(int i=0;i<n;i++){
1159             scanf("%d", &num[i]);
1160         }
1161         sort(num, num+n, cmp);
1162         for(int i=0;i<n;i++){
1163             bool ok=false;
1164             for(int j=0;j<n;j++){
1165                 if(i==j)continue;

```



```

1166         if(isprime[num[i]+num[j]]){
1167             ok=true;
1168             add(i, j);
1169         }
1170     }
1171     if(ok) sum++;
1172 }
1173 for(int i=0;i<n;i++) sort(g[i].begin(), g[i].end());
1174 if(sum<=k){
1175     printf("%d\n", sum);
1176 }
1177 else {
1178     int m=match();
1179     if(m>=k) printf("%d\n", 2*k);
1180     else if(sum-m<=k) printf("%d\n", sum);
1181     else printf("%d\n", k+m);
1182 }
1183 }
1184 }
1185
1186 //bzoj1010
1187 //决策单调性
1188 //
1189 //P教授要去看奥运，但是他舍不得他的玩具，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。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。
1190 //
1191 //input
1192 //5 4
1193 //3
1194 //4
1195 //2
1196 //1
1197 //4
1198 //
1199 //output
1200 //1
1201 //
1202 #include <bits/stdc++.h>
1203 using namespace std;
1204 typedef long long ll;
1205 const int maxn=50005;
1206 ll a[maxn], s[maxn], dp[maxn], l;
1207 int n;
1208 struct node{
1209     int l, r, p;
1210 }q[maxn];
1211 ll cal(int j, int i){
1212     return dp[j]+(s[i]-s[j]+i-j-1-l)*(s[i]-s[j]+i-j-1-l);
1213 }
1214 int bisearch(node a, int i){
1215     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
1216     while(l<=r){
1217         if(r-l<=1){
1218             if(cal(i, l)<cal(a.p, l)) tag=l;
1219             else tag=r;
1220             break;
1221         }
1222         int mid=l+r>>1;
1223         if(cal(i, mid)<cal(a.p, mid)) r=mid;
1224         else l=mid;
1225     }
1226     return tag;
1227 }
1228 void solve(){
1229     int head=1, tail=0;

```

```

1230     q[++tail]=(node){0, n, 0};
1231     for(int i=1;i<=n;i++){
1232         if(i>q[head].r)head++;
1233         dp[i]=cal(q[head].p, i);
1234         /*for(int i=head;i<=tail;i++){
1235             printf("node %d %d %d\n", q[i].l, q[i].r, q[i].p);
1236         }
1237         cout<<i<<' '<<q[head].p<<' '<<dp[i]<<endl;*/
1238         if(head>tail||cal(i, n)<cal(q[tail].p, n)){
1239             while(head<=tail&&cal(i, q[tail].l)<cal(q[tail].p, q[tail].l))tail--;
1240             if(head<=tail){
1241                 int t=bisearch(q[tail], i);
1242                 q[tail].r=t-1;
1243                 q[++tail]=(node){t, n, i};
1244             }
1245             else q[++tail]=(node){i, n, i};
1246         }
1247     }
1248 }
1249 int main(){
1250     scanf("%d%lld", &n, &l);
1251     for(ll i=1;i<=n;i++){
1252         scanf("%lld", &a[i]);
1253         s[i]=s[i-1]+a[i];
1254     }
1255     solve();
1256     printf("%lld\n", dp[n]);
1257 }
1258
1259
1260
1261 // #include <bitset>
1262 // using std::bitset;
1263 //bitset<n>b;
1264 //b.any() any 1?
1265 //b.none() no 1?
1266 //b.count() number of 1?
1267 //b.size() size?
1268 //b.set() set all pos 1
1269 //b.reset() set all pos 0
1270 //b.flip() 1->0, 0->1
1271
1272 //k中浓度为ai/1000的液体配成n/1000的浓度至少需要多少升
1273 //每种溶液均为整数升
1274 //易知若有解必小于1000, dp过程用bitset加速
1275
1276 #include <bits/stdc++.h>
1277 using namespace std;
1278
1279 int n, k;
1280 bool a[1050];
1281 bitset<2050>b[2];
1282
1283 int main(){
1284     scanf("%d%d", &n, &k);
1285     for(int i=1;i<=k;i++){
1286         int x;
1287         scanf("%d", &x);
1288         a[x]=true;
1289     }
1290     b[0][1000]=1;
1291     for(int i=0;i<=1000;i++){
1292         if(i!=0&&b[i%2][1000]){
1293             printf("%d\n", i);
1294             return 0;
1295         }
1296         int now=i%2;
1297         b[now^1].reset();
1298         for(int j=0;j<=1000;j++){
1299             if(a[j]){
1300                 b[now^1]|=(b[now]<<j)>>n;
1301             }
1302         }
1303     }

```

```

1303     }
1304     printf("-1\n");
1305 }
1306
1307
1308
1309 //好题
1310 //问题归结为:
1311 //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
1312 //用多重背包的按位拆分+bitset可以达到 $O(k \cdot \sigma(\log A_i) / 64)$ 
1313 //而由于 $\sigma(i \cdot A_i) = n$ , 最坏情况 $O(\sigma(\log A_i)) \sim O(\sqrt{n})$ 
1314 //所以总复杂度是 $O(k \cdot \sqrt{n} / 64)$ 
1315 #include <bits/stdc++.h>
1316 using namespace std;
1317
1318 int a[1050000];
1319 bool vis[1050000];
1320 int cnt[1050000];
1321 bitset<1050000> b;
1322 int n, k;
1323 int ma=0, mi, bonus=0;
1324
1325 int main() {
1326     scanf("%d%d", &n, &k);
1327     for(int i=1; i<=n; i++) scanf("%d", &a[i]);
1328     for(int i=1; i<=n; i++) {
1329         if(vis[i]) continue;
1330         int tmp=i;
1331         int len=0;
1332         while(!vis[tmp]) {
1333             vis[tmp]=true;
1334             len++;
1335             tmp=a[tmp];
1336         }
1337         cnt[len]++;
1338         bonus+=len/2;
1339     }
1340     ma=min(n, k+min(bonus, k));
1341     b[0]=1;
1342     for(int i=1; i<=n; i++) {
1343         if(!cnt[i]) continue;
1344         int k=1;
1345         while(k<cnt[i]) {
1346             b|=b<<k*i;
1347             cnt[i]-=k;
1348             k<<=1;
1349         }
1350         b|=b<<cnt[i]*i;
1351     }
1352     if(b[k]) mi=k;
1353     else mi=k+1;
1354     printf("%d %d\n", mi, ma);
1355 }
1356
1357 //树状数组区间修改区间查询
1358 //sum[n]=sigma(a[i])+sigma((n+1-i)d[i]);
1359 //其中a[i]是原数组元素
1360 //d[i]是[i, n]的共同增量
1361 #include <bits/stdc++.h>
1362 using namespace std;
1363 typedef long long ll;
1364 ll a[205000];
1365 ll d[205000];
1366 ll di[205000];
1367 int n, q;
1368
1369 int lowbit(int x) {
1370     return x&(-x);
1371 }
1372
1373 void update(int x, int num) {
1374     ll add=1LL*num;
1375     ll addi=1LL*num*x;

```

```

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

```

```

1449 }
1450 void geth(int n){
1451     int i, j, k=0;
1452     for(i=0;i<n;i++) Rank[sa[i]]=i;
1453     for(i=0;i<n;i++){
1454         if(k) k--;
1455         if(Rank[i]==0) break;
1456         j=sa[Rank[i]-1];
1457         while(s[i+k]==s[j+k]) k++;
1458         height[Rank[i]]=k;
1459     }
1460 }
1461
1462 int main()
1463 {
1464     n=8;
1465     s[8]=0;
1466     s[0]=s[1]=2;
1467     s[2]=s[3]=s[4]=s[5]=1;
1468     s[6]=s[7]=3;
1469     build_sa(200, n+1);
1470     geth(n+1);
1471     for(int i=0;i<=n;i++){
1472         printf("%d ", sa[i]);
1473     }
1474     printf("\n");
1475     for(int i=1;i<=n;i++){
1476         printf("%d ", height[i]);
1477     }
1478     printf("\n");
1479 }
1480
1481
1482
1483

```