

```

1  /* 数学 */
2  /* 扩展欧几里得 */
3  /* 素数线性筛 */
4  /* 中国剩余定理 */
5  /* 卢卡斯定理 */
6  /* 逆元 */
7  /* 欧拉函数 */
8  /* 莫比乌斯函数 */
9  /* 威佐夫博弈 */
10 /* 反NIM博弈 */
11 /* NTT/FFT/FWT */
12 /* Miller Rabin */
13 /* 线性递推  $m^{2\log n}$  */
14 /* polya定理 */
15
16 /* 数据结构 */
17 /* 单调队列 */
18 /* 吉司机线段树 */
19 /* 主席树 */
20 /* splay */
21 /* 树链剖分 */
22
23 /* 字符串 */
24 /* 后缀数组 */
25 /* AC自动机 */
26
27 /* 图论 */
28 /* lca */
29 /* tarjan */
30 /* 2-sat */
31 /* 网络流 */
32 /* KM */
33 /* 欧拉图 */
34 /* 点分治 */
35
36 /* dp */
37 /* 数位dp */
38 /* 斜率优化 */
39 /* 四边形优化 */
40 /* 决策单调性 */
41
42 /* 其他 */
43 /* 输入外挂 */
44 /* bitset */
45 /*  $O(1)$ 快速乘 */
46
47 //扩展欧几里得
48 //求 $ax+by=\gcd(a, b)$ 的解
49 void exgcd(ll a, ll b, ll &x, ll &y){
50     if(b==0){
51         x=1, y=0;
52         return ;
53     }
54     exgcd(b, a%b, x, y);
55     ll tmp=x;
56     x=y;
57     y=tmp-(a/b)*y;
58 }
59
60 //素数线性筛
61 const int maxn=205000;
62
63 bool vis[maxn];
64 int prime[maxn];
65 int tot;
66 void init(){
67     memset(vis, false, sizeof(vis));
68     tot=0;
69     for(int i=2;i<maxn;i++){
70         if(!vis[i])prime[++tot]=i;
71         for(int j=1;j<=tot&&prime[j]*i<maxn;j++){
72             vis[prime[j]*i]=true;
73             if(i%prime[j]==0)break;

```

```

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

```

```

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

```

```

220     }
221     if(on == -1)
222         for(int i = 0;i < len;i++)
223             y[i].r /= len;
224 }
225 const int MAXN = 200010;
226 complex x1[MAXN],x2[MAXN];
227 char str1[MAXN/2],str2[MAXN/2];
228 int sum[MAXN];
229 int main()
230 {
231     while(scanf("%s%s",str1,str2)==2)
232     {
233         int len1 = strlen(str1);
234         int len2 = strlen(str2);
235         int len = 1;
236         while(len < len1*2 || len < len2*2) len<<=1;
237         for(int i = 0;i < len1;i++)
238             x1[i] = complex(str1[len1-1-i]-'0',0);
239         for(int i = len1;i < len;i++)
240             x1[i] = complex(0,0);
241         for(int i = 0;i < len2;i++)
242             x2[i] = complex(str2[len2-1-i]-'0',0);
243         for(int i = len2;i < len;i++)
244             x2[i] = complex(0,0);
245         //求DFT
246         fft(x1,len,1);
247         fft(x2,len,1);
248         for(int i = 0;i < len;i++)
249             x1[i] = x1[i]*x2[i];
250         fft(x1,len,-1);
251         for(int i = 0;i < len;i++)
252             sum[i] = (int)(x1[i].r+0.5);
253         for(int i = 0;i < len;i++)
254         {
255             sum[i+1]+=sum[i]/10;
256             sum[i]%=10;
257         }
258         len = len1+len2-1;
259         while(sum[len] <= 0 && len > 0) len--;
260         for(int i = len;i >= 0;i--)
261             printf("%c",sum[i]+'0');
262         printf("\n");
263     }
264     return 0;
265 }
266
267 //NTT
268 #include<cstdio>
269 #include<cstring>
270 #include<algorithm>
271 using namespace std;
272 #define MAXN 262144
273
274 const long long P=50000000001507329LL; // 190734863287 * 2 ^ 18 + 1
275 //const int P=1004535809; // 479 * 2 ^ 21 + 1
276 //const int P=998244353; // 119 * 2 ^ 23 + 1
277 const int G=3;
278
279 long long mul(long long x,long long y)
280 {
281     return (x*y-(long long) (x/(long double) P*y+1e-3)*P+P)%P;
282 }
283 long long qpow(long long x,long long k,long long p)
284 {
285     long long ret=1;
286     while(k)
287     {
288         if(k&1) ret=mul(ret,x);
289         k>>=1;
290         x=mul(x,x);
291     }
292     return ret;

```

```

293 }
294
295 long long wn[25];
296 void getwn()
297 {
298     for(int i=1; i<=18; ++i)
299     {
300         int t=1<<i;
301         wn[i]=qpow(G, (P-1)/t, P);
302     }
303 }
304
305 int len;
306 void NTT(long long y[], int op)
307 {
308     for(int i=1, j=len>>1, k; i<len-1; ++i)
309     {
310         if(i<j) swap(y[i], y[j]);
311         k=len>>1;
312         while(j>=k)
313         {
314             j-=k;
315             k>>=1;
316         }
317         if(j<k) j+=k;
318     }
319     int id=0;
320     for(int h=2; h<=len; h<<=1)
321     {
322         ++id;
323         for(int i=0; i<len; i+=h)
324         {
325             long long w=1;
326             for(int j=i; j<i+(h>>1); ++j)
327             {
328                 long long u=y[j], t=mul(y[j+h/2], w);
329                 y[j]=u+t;
330                 if(y[j]>=P) y[j]-=P;
331                 y[j+h/2]=u-t+P;
332                 if(y[j+h/2]>=P) y[j+h/2]-=P;
333                 w=mul(w, wn[id]);
334             }
335         }
336     }
337     if(op== -1)
338     {
339         for(int i=1; i<len/2; ++i) swap(y[i], y[len-i]);
340         long long inv=qpow(len, P-2, P);
341         for(int i=0; i<len; ++i) y[i]=mul(y[i], inv);
342     }
343 }
344 void Convolution(long long A[], long long B[], int len1, int len2)
345 {
346     int n=max(len1, len2);
347     for(len=1; len<(n<<1); len<<=1);
348     for(int i=len1; i<len; ++i)
349     {
350         A[i]=0;
351     }
352     for (int i=len2; i<len; i++)
353     {
354         B[i]=0;
355     }
356
357     NTT(A, 1);
358     NTT(B, 1);
359     for(int i=0; i<len; ++i)
360     {
361         A[i]=mul(A[i], B[i]);
362     }
363     NTT(A, -1);
364 }
365

```

```

366 long long A[MAXN],B[MAXN];
367 char s1[MAXN],s2[MAXN];
368 void debug() {
369     A[0]=1, A[1]=2, A[2]=3;
370     B[0]=1, B[1]=2, B[2]=3;
371     Convolution(A, B, 3, 3);
372     for(int i=0;i<=6;i++)printf("%lld\n", A[i]);
373 }
374 int main()
375 {
376     getwn();
377     debug();
378 }
379
380 //FWT
381 //rm to do
382
383 //Miller Rabin
384 //Pollard_rho
385
386 #include <bits/stdc++.h>
387 using namespace std;
388
389 typedef unsigned long long ll;
390
391 const int T=10;
392
393 ll mul(ll x,ll y, ll P)
394 {
395     return (x*y-(ll)(x/(ll)P*y+1e-3)*P+P)%P;
396 }
397
398 ll exp(ll x, ll y, ll mod){
399     ll ans=1;
400     ll base=x;
401     while(y){
402         if(y&1){
403             ans=mul(ans, base, mod);
404         }
405         y>>=1;
406         base=mul(base, base, mod);
407     }
408     return ans;
409 }
410
411 bool miller_rabin(ll n)
412 {
413     if(n==2)return true;
414     if(n<2||!(n&1))return false;
415     ll m=n-1;
416     ll k=0;
417     while(!(m&1)){
418         k++;
419         m>>=1;
420     }
421     for(int i=0;i<T;i++){
422         ll a=rand()%(n-1)+1;
423         ll x=exp(a, m, n);
424         ll y=0;
425         for(int j=1;j<=k;j++){
426             y=mul(x, x, n);
427             if(y==1&&x!=1&&x!=n-1)return false;
428             x=y;
429         }
430         if(y!=1)return false;
431     }
432     return true;
433 }
434
435 ll Pollard_rho(ll x,ll c){
436     ll i=1,k=2;
437     ll x0=rand()%x;
438     ll y=x0;

```

```

439     while(1) {
440         i++;
441         x0=(mul(x0,x0,x)+c)%x;
442         ll d=__gcd(y-x0,x);
443         if(d!=1&&d!=x) return d;
444         if(y==x0) return x;
445         if(i==k){y=x0;k+=k;}
446     }
447 }
448
449 void debug() {
450     ll n;
451     while(cin>>n) {
452         //if(miller_rabin(n)) cout<<"YES\n";
453         //else cout<<"NO\n";
454         cout<<Pollard_rho(n, rand()%(n-1)+1)<<endl;
455     }
456 }
457
458 int main() {
459     srand(time(0));
460     debug();
461 }
462
463
464 //AC自动机
465 //查询模式串出现次数
466 #include <bits/stdc++.h>
467 using namespace std;
468 struct Trie{
469     int ch[50500][27];
470     int fail[50500];
471     int end[50500];
472     int sz;
473     int newnode() {
474         for(int i=0;i<27;i++) ch[sz][i]=-1;
475         end[sz]=-1;
476         return sz++;
477     }
478     void init() {
479         sz=0;
480         newnode();
481     }
482     int idx(int c){
483         if(c>='A'&&c<='Z') return c-'A';
484         return 26;
485     }
486     void insert(char s[], int id){
487         int len=strlen(s);
488         int u=0;
489         for(int i=0;i<len;i++){
490             int c=idx(s[i]);
491             if(ch[u][c]==-1){
492                 ch[u][c]=newnode();
493             }
494             u=ch[u][c];
495         }
496         end[u]=id;
497     }
498     void build() {
499         queue<int>q;
500         fail[0]=0;
501         for(int i=0;i<27;i++){
502             if(ch[0][i]==-1)ch[0][i]=0;
503             else {
504                 fail[ch[0][i]]=0;
505                 q.push(ch[0][i]);
506             }
507         }
508         while(!q.empty()){
509             int u=q.front();
510             q.pop();

```

```

512         for(int i=0;i<27;i++){
513             if(ch[u][i]==-1){
514                 ch[u][i]=ch[fail[u]][i];
515             }
516             else {
517                 fail[ch[u][i]]=ch[fail[u]][i];
518                 q.push(ch[u][i]);
519             }
520         }
521     }
522 }
523 };
524
525 Trie T;
526 char s[1050][55];
527 char str[2050000];
528 int cnt[1050];
529 int n;
530
531 int main()
532 {
533     while(~scanf("%d", &n)){
534         memset(cnt, 0, sizeof(cnt));
535         T.init();
536         for(int i=1;i<=n;i++){
537             scanf("%s", s[i]);
538             T.insert(s[i], i);
539         }
540         T.build();
541         scanf("%s", str);
542         int len=strlen(str);
543         int u=0;
544         for(int i=0;i<len;i++){
545             int c=T.idx(str[i]);
546             u=T.ch[u][c];
547             int tmp=u;
548             while(tmp){
549                 if(T.end[tmp]!=-1){
550                     cnt[T.end[tmp]]++;
551                 }
552                 tmp=T.fail[tmp];
553             }
554         }
555         for(int i=1;i<=n;i++){
556             if(cnt[i]){
557                 printf("%s: %d\n", s[i], cnt[i]);
558             }
559         }
560     }
561 }
562
563 //数位DP
564 // pos    = 当前处理的位置(一般从高位到低位)
565 // pre    = 上一个位的数字(更高的那一位)
566 // 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,
567 //         给计数器+1。
568 // 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,
569 //         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
570 //         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
571 //         话此时的limit=1,也就是说当前只可以取0-6。
572 //
573 // 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。
574 int dfs(int pos,int pre,int status,int limit)
575 {
576     //已搜到尽头,返回"是否找到了答案"这个状态。
577     if(pos < 1)
578         return status;
579
580     //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。
581     if(!limit && DP[pos][pre][status] != -1)
582         return DP[pos][pre][status];
583
584     int end = limit ? DIG[pos] : 9;

```



```

585     int    ret = 0;
586
587     //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
588     //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
589     //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
590     //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。
591     //
592     //这里用"不要49"一题来做例子。
593     for(int i = 0;i <= end;i ++){
594         ret += dfs(pos - 1,i,status || (pre == 4 && i == 9),limit && (i == end));
595     }
596     //DP里保存完整的、取到尽头的数据
597     if(!limit)
598         DP[pos][pre][status] = ret;
599
600     return    ret;
601 }
602
603 //点分治
604 #include <cstdio>
605 #include <cstring>
606 #include <algorithm>
607 #include <vector>
608 using namespace std;
609 typedef long long ll;
610
611 struct Edge{
612     int to, next, len;
613 }edge[20500];
614
615 int head[20500];
616 bool vis[20500];
617 int siz[20500];
618 int f[20500];
619 int cnt, rt, sum, n, k;
620 ll ans;
621 ll vs[20500];
622 int vsc;
623 ll v[20500];
624 int vc;
625
626 void init(){
627     memset(head, -1, sizeof(head));
628     memset(vis, false, sizeof(vis));
629     cnt=0;
630     ans=0;
631 }
632
633 void add(int u, int v, int w){
634     edge[cnt].to=v;
635     edge[cnt].len=w;
636     edge[cnt].next=head[u];
637     head[u]=cnt++;
638 }
639
640 void getrt(int u, int fa){
641     siz[u]=1;
642     f[u]=0;
643     for(int i=head[u];~i;i=edge[i].next){
644         if(edge[i].to!=fa&&!vis[edge[i].to]){
645             getrt(edge[i].to, u);
646             siz[u]+=siz[edge[i].to];
647             f[u]=max(f[u], siz[edge[i].to]);
648         }
649     }
650     f[u]=max(f[u], sum-siz[u]);
651     if(f[u]<f[rt])rt=u;
652 }
653
654 void getdeep(int u, int fa, ll len){
655     vs[++vsc]=len;
656     v[++vc]=len;
657     for(int i=head[u];~i;i=edge[i].next){

```

```

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

```

```

731 ll n;
732 int u, d;
733 int a[100], b[100];
734 ll dp[205];
735 ll A[205];
736 ll C[205];
737
738 // given first m a[i] and coef (0 based)
739 // calc a[n] % MOD in O(m*m*log(n))
740 // a[n] = sum(c[m - i] * a[n - i]) i = 1....m
741 // a[m] = sum(c[i] * a[i]), i = 0.....m-1
742
743 LL linear_recurrence(LL n, int m, LL a[], LL c[], int p){ //n->a[i], m -> c[i]
744     LL v[MAXN] = {1 % MOD}, u[MAXN << 1], msk = !!n;
745     for(LL i = n; i > 1; i >>= 1) msk <<= 1;
746     for(LL x = 0; msk; msk >>= 1, x <<= 1){
747         fill_n(u, m << 1, 0);
748         int b = !(n & msk); x |= b;
749         if(x < m) u[x] = 1 % p;
750         else{
751             for(int i = 0; i < m; i++){
752                 for(int j = 0, t = i + b; j < m; ++j, ++t)
753                     u[t] = (u[t] + v[i] * v[j]) % MOD;
754             }
755             for(int i = (m << 1) - 1; i >= m; --i){
756                 for(int j = 0, t = i - m; j < m; ++j, ++t){
757                     u[t] = (u[t] + c[j] * u[i]) % MOD;
758                 }
759             }
760         }
761         copy(u, u+m, v);
762     }
763     LL ans = 0;
764     for(int i = 0; i < m; i++){
765         ans = (ans + v[i] * a[i]) % MOD;
766     }
767     return ans;
768 }
769
770 bool vis[205];
771 int main()
772 {
773     while(~scanf("%lld", &n)){
774         int ma=0;
775         memset(dp, 0, sizeof(dp));
776         memset(A, 0, sizeof(A));
777         memset(C, 0, sizeof(C));
778         memset(vis, false, sizeof(vis));
779         scanf("%d", &u);
780         for(int i=1;i<=u;i++){scanf("%d", &a[i]);}
781         scanf("%d", &d);
782         for(int i=1;i<=d;i++){scanf("%d", &b[i]);ma=max(b[i], ma);vis[b[i]]=true;}
783         dp[0]=1;
784         for(int i=1;i<=ma;i++){
785             for(int j=1;j<=u;j++){
786                 if(i<a[j])continue;
787                 dp[i]=(dp[i-a[j]]+dp[i])%mod;
788             }
789         }
790         vis[0]=true;
791         for(int i=0;i<=ma;i++)if(!vis[i])dp[i]=0;
792         for(int i=0;i<=ma;i++){C[i]=dp[ma-i];}
793         A[0]=1;
794         for(int i=1;i<=ma;i++){
795             for(int j=1;j<=i;j++){
796                 A[i]=(A[i]+A[i-j]*dp[j])%mod;
797             }
798         }
799         ll ans=linear_recurrence(n, ma, A, C, mod);
800         printf("%lld\n", ans);
801     }
802 }
803

```

```

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

```

```

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

```

```

948         printf("%d\n", getmx(1, x, y, 1, n));
949     }
950     else {
951         int x, y;
952         scanf("%d%d", &x, &y);
953         printf("%lld\n", getsum(1, x, y, 1, n));
954     }
955 }
956 }
957 }
958
959 //树链剖分
960 //input:
961 //1
962 //3
963 //1 2 1
964 //2 3 2
965 //QUERY 1 2
966 //CHANGE 1 3
967 //QUERY 1 2
968 //DONE
969 //
970 //Output:
971 //1
972 //3
973
974 #include <bits/stdc++.h>
975 using namespace std;
976 const int maxn=10500;
977
978 int dep[maxn],siz[maxn],fa[maxn],id[maxn],son[maxn],val[maxn],top[maxn];
979 int num,n;
980 int ma[maxn<<2];
981 vector<int>g[maxn];
982 struct edge{
983     int x,y,val;
984     void read(){
985         scanf("%d%d%d",&x,&y,&val);
986     }
987 }e[maxn];
988
989 void init(){
990     for(int i=1;i<=n;i++)g[i].clear();
991 }
992
993 void dfs1(int u,int f,int d){
994     dep[u]=d,siz[u]=1,son[u]=0,fa[u]=f;
995     for(int i=0;i<g[u].size();i++){
996         int v=g[u][i];
997         if(v==f)continue;
998         dfs1(v,u,d+1);
999         siz[u]+=siz[v];
1000         if(siz[son[u]]<siz[v])son[u]=v;
1001     }
1002 }
1003
1004 void dfs2(int u,int tp){
1005     top[u]=tp;
1006     id[u]=++num;
1007     if(son[u])dfs2(son[u],tp);
1008     for(int i=0;i<g[u].size();i++){
1009         int v=g[u][i];
1010         if(v==fa[u]||v==son[u])continue;
1011         dfs2(v,v);
1012     }
1013 }
1014
1015 void pushup(int u){
1016     ma[u]=max(ma[2*u],ma[2*u+1]);
1017 }
1018
1019 void build(int u,int l,int r){
1020     if(l==r){

```

```

1021     ma[u]=val[l];
1022     return ;
1023 }
1024 int mid=(l+r)/2;
1025 build(2*u,l,mid);
1026 build(2*u+1,mid+1,r);
1027 pushup(u);
1028 }
1029
1030 void update(int u,int p,int l,int r,int q){
1031     if(p<l||p>r) return;
1032     if(l==r){
1033         ma[u]=q;
1034         return;
1035     }
1036     int mid=(l+r)/2;
1037     update(2*u,p,l,mid,q);
1038     update(2*u+1,p,mid+1,r,q);
1039     pushup(u);
1040 }
1041
1042 int query(int u,int ql,int qr,int l,int r){
1043     if(ql>r||qr<l) return 0;
1044     if(ql<=l&&r<=qr) return ma[u];
1045     int mid=(l+r)/2;
1046     return max(query(2*u,ql,qr,l,mid),query(2*u+1,ql,qr,mid+1,r));
1047 }
1048
1049
1050 int Query(int u,int v){
1051     int ret=0;
1052     int tp1=top[u],tp2=top[v];
1053     while(tp1!=tp2){
1054         if(dep[tp1]<dep[tp2]){
1055             swap(u,v);swap(tp1,tp2);
1056         }
1057         ret=max(ret,query(1,id[tp1],id[u],1,num));
1058         u=fa[tp1];
1059         tp1=top[u];
1060     }
1061     if(u==v) return ret;
1062     if(dep[u]<dep[v]) swap(u,v);
1063     ret=max(ret,query(1,id[son[v]],id[u],1,num));
1064     return ret;
1065 }
1066
1067
1068
1069
1070 int main(){
1071     int T;
1072     scanf("%d",&T);
1073     while(T--){
1074         scanf("%d",&n);
1075         init();
1076         for(int i=1;i<n;i++){
1077             e[i].read();
1078             g[e[i].x].push_back(e[i].y);
1079             g[e[i].y].push_back(e[i].x);
1080         }
1081         num=0;
1082         dfs1(1,0,1);
1083         dfs2(1,1);
1084         for(int i=1;i<n;i++){
1085             if(dep[e[i].x]<dep[e[i].y]) swap(e[i].x,e[i].y);
1086             val[id[e[i].x]]=e[i].val;
1087         }
1088         build(1,1,num);
1089         char tag[30];
1090         while(scanf("%s",tag)){
1091             if(tag[0]=='D') break;
1092             if(tag[0]=='Q'){
1093                 int x,y;

```

```

1094         scanf("%d%d",&x,&y);
1095         printf("%d\n",Query(x,y));
1096     }
1097     else {
1098         int pos,x;
1099         scanf("%d%d",&pos,&x);
1100         update(1,id[e[pos].x],1,num,x);
1101     }
1102 }
1103 }
1104 }
1105
1106 //bzoj1010
1107 //决策单调性
1108 //
1109 //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
 $x=j-i+\text{Sigma}(C_k) \quad i \leq K \leq j$ 
制作容器的费用与容器的长度有关，根据教授研究，如果容器长度为x,其制作费用为 $(x-L)^2$ .其中
L是一个常量。P教授不关心容器的数目，他可以制作出任意长度的容器，甚至超过L。但他希望费
用最。
1110 //
1111 //input
1112 //5 4
1113 //3
1114 //4
1115 //2
1116 //1
1117 //4
1118 //
1119 //output
1120 //1
1121 //
1122 #include <bits/stdc++.h>
1123 using namespace std;
1124 typedef long long ll;
1125 const int maxn=50005;
1126 ll a[maxn], s[maxn], dp[maxn], l;
1127 int n;
1128 struct node{
1129     int l, r, p;
1130 }q[maxn];
1131 ll cal(int j, int i){
1132     return dp[j]+(s[i]-s[j]+i-j-1-l)*(s[i]-s[j]+i-j-1-l);
1133 }
1134 int bisearch(node a, int i){
1135     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
1136     while(l<=r){
1137         if(r-l<=1){
1138             if(cal(i, l)<cal(a.p, l))tag=l;
1139             else tag=r;
1140             break;
1141         }
1142         int mid=l+r>>1;
1143         if(cal(i, mid)<cal(a.p, mid))r=mid;
1144         else l=mid;
1145     }
1146     return tag;
1147 }
1148 void solve(){
1149     int head=1, tail=0;
1150     q[++tail]=(node){0, n, 0};
1151     for(int i=1;i<=n;i++){
1152         if(i>q[head].r)head++;
1153         dp[i]=cal(q[head].p, i);
1154         /*for(int i=head;i<=tail;i++){
1155             printf("node %d %d %d\n", q[i].l, q[i].r, q[i].p);
1156         }
1157         cout<<i<< ' '<<q[head].p<< ' '<<dp[i]<<endl;*/

```



```

1158         if(head>tail||cal(i, n)<cal(q[tail].p, n)){
1159             while(head<=tail&&cal(i, q[tail].l)<cal(q[tail].p, q[tail].l))tail--;
1160             if(head<=tail){
1161                 int t=bisearch(q[tail], i);
1162                 q[tail].r=t-1;
1163                 q[++tail]=(node){t, n, i};
1164             }
1165             else q[++tail]=(node){i, n, i};
1166         }
1167     }
1168 }
1169 int main(){
1170     scanf("%d%lld", &n, &l);
1171     for(ll i=1;i<=n;i++){
1172         scanf("%lld", &a[i]);
1173         s[i]=s[i-1]+a[i];
1174     }
1175     solve();
1176     printf("%lld\n", dp[n]);
1177 }
1178
1179
1180
1181 // #include <bitset>
1182 // using std::bitset;
1183 //bitset<n>b;
1184 //b.any() any 1?
1185 //b.none() no 1?
1186 //b.count() number of 1?
1187 //b.size() size?
1188 //b.set() set all pos 1
1189 //b.reset() set all pos 0
1190 //b.flip() 1->0, 0->1
1191
1192 //k中浓度为ai/1000的液体配成n/1000的浓度至少需要多少升
1193 //每种溶液均为整数升
1194 //易知若有解必小于1000, dp过程用bitset加速
1195
1196 #include <bits/stdc++.h>
1197 using namespace std;
1198
1199 int n, k;
1200 bool a[1050];
1201 bitset<2050>b[2];
1202
1203 int main(){
1204     scanf("%d%d", &n, &k);
1205     for(int i=1;i<=k;i++){
1206         int x;
1207         scanf("%d", &x);
1208         a[x]=true;
1209     }
1210     b[0][1000]=1;
1211     for(int i=0;i<=1000;i++){
1212         if(i!=0&&b[i%2][1000]){
1213             printf("%d\n", i);
1214             return 0;
1215         }
1216         int now=i%2;
1217         b[now^1].reset();
1218         for(int j=0;j<=1000;j++){
1219             if(a[j]){
1220                 b[now^1]|=(b[now]<<j)>>n;
1221             }
1222         }
1223     }
1224     printf("-1\n");
1225 }
1226
1227 //树状数组区间修改区间查询
1228 //sum[n]=sigma(a[i])+sigma((n+1-i)d[i]);
1229 //其中a[i]是原数组元素
1230 //d[i]是[i, n]的共同增量

```

```

1231 #include <bits/stdc++.h>
1232 using namespace std;
1233 typedef long long ll;
1234 ll a[205000];
1235 ll d[205000];
1236 ll di[205000];
1237 int n, q;
1238
1239 int lowbit(int x){
1240     return x&(-x);
1241 }
1242
1243 void update(int x, int num){
1244     ll add=1LL*num;
1245     ll addi=1LL*num*x;
1246     while(x<=n){
1247         d[x]+=add;
1248         di[x]+=addi;
1249         x+=lowbit(x);
1250     }
1251 }
1252
1253 ll query(int x){
1254     ll ans=a[x];
1255     ll tmp=x;
1256     while(tmp){
1257         ans+=d[tmp]*(x+1);
1258         ans+=di[tmp]*(-1);
1259         tmp-=lowbit(tmp);
1260     }
1261     return ans;
1262 }
1263
1264 int main()
1265 {
1266     scanf("%d", &n);
1267     for(int i=1;i<=n;i++){
1268         scanf("%lld", &a[i]);
1269         a[i]+=a[i-1];
1270     }
1271     scanf("%d", &q);
1272     while(q--){
1273         int tag;
1274         scanf("%d", &tag);
1275         if(tag==1){
1276             int x, y, num;
1277             scanf("%d%d%d", &x, &y, &num);
1278             update(x, num);
1279             update(y+1, -num);
1280         }
1281         else {
1282             int x, y;
1283             scanf("%d%d", &x, &y);
1284             printf("%lld\n", query(y)-query(x-1));
1285         }
1286     }
1287 }
1288
1289

```