

```

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
22 /* 字符串 */
23 /* 后缀数组 */
24 /* AC自动机 */
25
26 /* 图论 */
27 /* lca */
28 /* tarjan */
29 /* 2-sat */
30 /* 网络流 */
31 /* KM */
32 /* 欧拉图 */
33 /* 点分治 */
34
35 /* dp */
36 /* 数位dp */
37 /* 斜率优化 */
38 /* 四边形优化 */
39
40 /* 其他 */
41 /* 输入外挂 */
42 /* bitset */
43 /*  $O(1)$ 快速乘 */
44
45 //扩展欧几里得
46 //求 $ax+by=\gcd(a, b)$ 的解
47 void exgcd(ll a, ll b, ll &x, ll &y){
48     if(b==0){
49         x=1, y=0;
50         return ;
51     }
52     exgcd(b, a%b, x, y);
53     ll tmp=x;
54     x=y;
55     y=tmp-(a/b)*y;
56 }
57
58 //素数线性筛
59 const int maxn=205000;
60
61 bool vis[maxn];
62 int prime[maxn];
63 int tot;
64 void init(){
65     memset(vis, false, sizeof(vis));
66     tot=0;
67     for(int i=2;i<maxn;i++){
68         if(!vis[i])prime[++tot]=i;
69         for(int j=1;j<=tot&&prime[j]*i<maxn;j++){
70             vis[prime[j]*i]=true;
71             if(i%prime[j]==0)break;
72         }
73     }

```

```

74 }
75
76 //中国剩余定理
77 //正整数m1, m2, ..., mk两两互素, 则同余方程组
78 //
79 //x%m1==a1
80 //x%m2==a2
81 //
82 //x%mk==ak
83 //
84 //在模M=m1*m1*...*mk下的解唯一
85 //
86 //x%M==sigma(ai*Mi*inv(Mi))
87 //
88 //Mi=M/mi, inv(Mi) 为Mi模mi的逆元
89 int CRT(int a[], int m[], int n){
90     int M=1;
91     int ans=0;
92     for(int i=1;i<=n;i++) M*=m[i];
93     for(int i=1;i<=n;i++){
94         int x, y;
95         int Mi=M/m[i];
96         exgcd(Mi, m[i], x, y); //m[i]不一定是素数, 故要用exgcd求逆元
97         ans=(ans+a[i]*Mi*x)%M;
98     }
99     if(ans<0) ans+=M;
100     return ans;
101 }
102
103 //卢卡斯定理
104 //rm to do
105
106 //逆元
107 //ans=(a/b)%m=(a%(mb))/b
108 //fermat小定理
109 //exgcd
110 //o(n) 预处理mod的逆元
111 int mod;
112 int inv[maxn];
113
114 void init(){
115     inv[1] = 1;
116     for(int i=2;i<mod;i++){
117         inv[i]=inv[mod%i]*(mod-mod/i)%mod;
118     }
119 }
120
121 //欧拉函数
122 //rm to do
123
124 //莫比乌斯函数
125 //rm to do
126
127 //威佐夫博弈
128 //rm to do
129
130 //反NIM博弈
131 //rm to do
132
133 //FFT
134 #include <stdio.h>
135 #include <string.h>
136 #include <iostream>
137 #include <algorithm>
138 #include <math.h>
139 using namespace std;
140
141 const double PI = acos(-1.0);
142 //复数结构体
143 struct complex
144 {
145     double r,i;
146     complex(double _r = 0.0,double _i = 0.0)

```

```

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

```

```

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

```

```

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

```

```

366 }
367
368 //FWT
369 //rm to do
370
371 //Miller Rabin
372
373 #include <bits/stdc++.h>
374 using namespace std;
375
376 typedef unsigned long long ll;
377
378 const int T=10;
379
380 ll exp(ll x, ll y, ll mod){
381     ll ans=1;
382     ll base=x;
383     while(y){
384         if(y&1){
385             ans=ans*base%mod;
386         }
387         y>>=1;
388         base=base*base%mod;
389     }
390     return ans;
391 }
392
393 bool miller_rabin(ll n)
394 {
395     if(n==2) return true;
396     if(n<2||!(n&1)) return false;
397     ll m=n-1;
398     ll k=0;
399     while(!(m&1)){
400         k++;
401         m>>=1;
402     }
403     for(int i=0;i<T;i++){
404         ll a=rand()%(n-1)+1;
405         ll x=exp(a, m, n);
406         ll y=0;
407         for(int j=1;j<=k;j++){
408             y=x*x%n;
409             if(y==1&&x!=1&&x!=n-1) return false;
410             x=y;
411         }
412         if(y!=1) return false;
413     }
414     return true;
415 }
416
417 void debug(){
418     int n;
419     while(cin>>n){
420         if(miller_rabin(n)) cout<<"YES\n";
421         else cout<<"NO\n";
422     }
423 }
424
425 int main(){
426     srand(time(0));
427     debug();
428 }
429
430 //后缀数组
431 #include <cstdio>
432 #include <cstring>
433 #include <algorithm>
434 using namespace std;
435 const int maxn=20500;
436 int s[maxn];
437 int sa[maxn], t[maxn], t2[maxn], c[maxn], n;
438 int Rank[maxn], height[maxn];

```

```

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

```

```

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

```



```

585     }
586 }
587 }
588 }
589
590 //数位DP
591 // pos = 当前处理的位置(一般从高位到低位)
592 // pre = 上一个位的数字(更高的那一位)
593 // 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,
594 // 给计数器+1。
595 // 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,
596 // 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
597 // 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
598 // 话此时的limit=1,也就是说当前只可以取0-6。
599 //
600 // 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。
601 int dfs(int pos,int pre,int status,int limit)
602 {
603     //已搜到尽头,返回"是否找到了答案"这个状态。
604     if(pos < 1)
605         return status;
606
607     //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。
608     if(!limit && DP[pos][pre][status] != -1)
609         return DP[pos][pre][status];
610
611     int end = limit ? DIG[pos] : 9;
612     int ret = 0;
613
614     //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
615     //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
616     //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
617     //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。
618     //
619     //这里用"不要49"一题来做例子。
620     for(int i = 0;i <= end;i ++)
621         ret += dfs(pos - 1,i,status || (pre == 4 && i == 9),limit && (i == end));
622
623     //DP里保存完整的、取到尽头的数据
624     if(!limit)
625         DP[pos][pre][status] = ret;
626
627     return ret;
628 }
629
630 //点分治
631 #include <cstdio>
632 #include <cstring>
633 #include <algorithm>
634 #include <vector>
635 using namespace std;
636 typedef long long ll;
637
638 struct Edge{
639     int to, next, len;
640 }edge[20500];
641
642 int head[20500];
643 bool vis[20500];
644 int siz[20500];
645 int f[20500];
646 int cnt, rt, sum, n, k;
647 ll ans;
648 ll vs[20500];
649 int vsc;
650 ll v[20500];
651 int vc;
652
653 void init(){
654     memset(head, -1, sizeof(head));
655     memset(vis, false, sizeof(vis));
656     cnt=0;
657     ans=0;

```

```

658 }
659
660 void add(int u, int v, int w){
661     edge[cnt].to=v;
662     edge[cnt].len=w;
663     edge[cnt].next=head[u];
664     head[u]=cnt++;
665 }
666
667 void getrt(int u, int fa){
668     siz[u]=1;
669     f[u]=0;
670     for(int i=head[u];~i;i=edge[i].next){
671         if(edge[i].to!=fa&&!vis[edge[i].to]){
672             getrt(edge[i].to, u);
673             siz[u]+=siz[edge[i].to];
674             f[u]=max(f[u], siz[edge[i].to]);
675         }
676     }
677     f[u]=max(f[u], sum-siz[u]);
678     if(f[u]<f[rt])rt=u;
679 }
680
681 void getdeep(int u, int fa, ll len){
682     vs[++vsc]=len;
683     v[++vc]=len;
684     for(int i=head[u];~i;i=edge[i].next){
685         if(edge[i].to!=fa&&!vis[edge[i].to]){
686             getdeep(edge[i].to, u, len+edge[i].len);
687         }
688     }
689 }
690
691 void solve(int u){
692     vis[u]=true;
693     vsc=0;
694     for(int i=head[u];~i;i=edge[i].next){
695         if(!vis[edge[i].to]){
696             vc=0;
697             getdeep(edge[i].to, u, edge[i].len);
698             sort(v+1, v+1+vc);
699             int l=1, r=vc;
700             while(l<r){
701                 if(v[l]+v[r]<=k){
702                     ans+=(r-l);
703                     l++;
704                 }
705                 else r--;
706             }
707         }
708     }
709     sort(vs+1, vs+1+vsc);
710     int l=1, r=vsc;
711     while(l<r){
712         if(vs[l]+vs[r]<=k){
713             ans+=(r-l);
714             l++;
715         }
716         else r--;
717     }
718     for(int i=1;i<=vsc;i++){
719         if(vs[i]<=k)ans++;
720     }
721     for(int i=head[u];~i;i=edge[i].next){
722         if(!vis[edge[i].to]){
723             rt=0, sum=siz[edge[i].to];
724             getrt(edge[i].to, 0);
725             solve(rt);
726         }
727     }
728 }
729
730 int main()

```

```

731 {
732     while(~scanf("%d%d", &n, &k)){
733         if(!n&&!k)break;
734         init();
735         for(int i=1;i<n;i++){
736             int u, v, w;
737             scanf("%d%d%d", &u, &v, &w);
738             add(u, v, w);
739             add(v, u, w);
740         }
741         f[0]=30000;
742         rt=0, sum=n;
743         getrt(1, 0);
744         solve(rt);
745         printf("%lld\n", ans);
746     }
747 }
748
749 //线性递推模板m^2logn
750 #include <bits/stdc++.h>
751 #pragma comment(linker, "/STACK:102400000,102400000")
752 using namespace std;
753 typedef long long ll;
754 #define LL long long
755 const int mod=1e9+7;
756 const int MOD=1e9+7;
757 const int MAXN=205;
758 ll n;
759 int u, d;
760 int a[100], b[100];
761 ll dp[205];
762 ll A[205];
763 ll C[205];
764
765 // given first m a[i] and coef (0 based)
766 // calc a[n] % MOD in O(m*m*log(n))
767 // a[n] = sum(c[m - i] * a[n - i]) i = 1....m
768 // a[m] = sum(c[i] * a[i]), i = 0.....m-1
769
770 LL linear_recurrence(LL n, int m, LL a[], LL c[], int p){ //n->a[i], m -> c[i]
771     LL v[MAXN] = {1 % MOD}, u[MAXN << 1], msk = !!n;
772     for(LL i = n; i > 1; i >>= 1) msk <<= 1;
773     for(LL x = 0; msk; msk >>= 1, x <<= 1){
774         fill_n(u, m << 1, 0);
775         int b = !(n & msk); x |= b;
776         if(x < m) u[x] = 1 % p;
777         else{
778             for(int i = 0; i < m; i++){
779                 for(int j = 0, t = i + b; j < m; ++j, ++t)
780                     u[t] = (u[t] + v[i] * v[j]) % MOD;
781             }
782             for(int i = (m << 1) - 1; i >= m; --i){
783                 for(int j = 0, t = i - m; j < m; ++j, ++t){
784                     u[t] = (u[t] + c[j] * u[i]) % MOD;
785                 }
786             }
787         }
788         copy(u, u+m, v);
789     }
790     LL ans = 0;
791     for(int i = 0; i < m; i++){
792         ans = (ans + v[i] * a[i]) % MOD;
793     }
794     return ans;
795 }
796
797 bool vis[205];
798 int main()
799 {
800     while(~scanf("%lld", &n)){
801         int ma=0;
802         memset(dp, 0, sizeof(dp));
803         memset(A, 0, sizeof(A));

```

```

804     memset(C, 0, sizeof(C));
805     memset(vis, false, sizeof(vis));
806     scanf("%d", &u);
807     for(int i=1;i<=u;i++){scanf("%d", &a[i]);}
808     scanf("%d", &d);
809     for(int i=1;i<=d;i++){scanf("%d", &b[i]);ma=max(b[i], ma);vis[b[i]]=true;}
810     dp[0]=1;
811     for(int i=1;i<=ma;i++){
812         for(int j=1;j<=u;j++){
813             if(i<a[j]) continue;
814             dp[i]=(dp[i-a[j]]+dp[i])%mod;
815         }
816     }
817     vis[0]=true;
818     for(int i=0;i<=ma;i++) if(!vis[i]) dp[i]=0;
819     for(int i=0;i<ma;i++){C[i]=dp[ma-i];}
820     A[0]=1;
821     for(int i=1;i<ma;i++){
822         for(int j=1;j<=i;j++){
823             A[i]=(A[i]+A[i-j]*dp[j])%mod;
824         }
825     }
826     ll ans=linear_recurrence(n, ma, A, C, mod);
827     printf("%lld\n", ans);
828 }
829 }
830
831 //吉司机线段树
832 void update(int u, int ql, int qr, int c, int l, int r){
833     if(r<ql||qr<l||cut()) return;
834     if(ql<=l&&r<=qr&&check()){
835         putlazy(u,c);
836         return;
837     }
838     int mid=(l+r)/2;
839     pushdown(u);
840     update(2*u, ql, qr, c, l, mid);
841     update(2*u+1, ql, qr, c, mid+1, r);
842     pushup(u);
843 }
844
845 //对于hdu5306这个题，考虑区间与t取min这个操作，有如下几种情况：
846 //case 1: t大于最大值，此时区间不变；
847 //case
848 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
849 ;
850 //case 3: t大于严格次大值，小于最大值，这里可以打懒标记。
851 //
852 //考虑查询，只需维护最大值，最大值个数，严格次大值即可。
853 //吉司机宇宙线段树之王！
854 #include <bits/stdc++.h>
855 using namespace std;
856 const int maxn=1000005;
857 typedef long long ll;
858
859 int mx[maxn<<2];
860 int cnt[maxn<<2];
861 int se[maxn<<2];
862 int lazy[maxn<<2];
863 ll sum[maxn<<2];
864 int a[maxn];
865 int n, m;
866
867 void putlazy(int u, int t){
868     sum[u]-=1LL*cnt[u]*(mx[u]-t);
869     mx[u]=t;
870     lazy[u]=t;
871 }
872
873 void pushdown(int u){
874     if(lazy[u]==-1) return;
875     if(mx[2*u]>lazy[u]){
876         sum[2*u]-=1LL*cnt[2*u]*(mx[2*u]-lazy[u]);

```

```

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

```

```

948     if(ql<=l&&r<=qr) return sum[u];
949     pushdown(u);
950     int mid=l+r>>1;
951     ll ans=0;
952     ans+=getsum(2*u, ql, qr, l, mid);
953     ans+=getsum(2*u+1, ql, qr, mid+1, r);
954     return ans;
955 }
956
957 int main(){
958     int T;
959     scanf("%d", &T);
960     while(T--){
961         scanf("%d%d", &n, &m);
962         for(int i=1;i<=n;i++) scanf("%d", &a[i]);
963         build(1, 1, n);
964         for(int i=1;i<=m;i++){
965             int tag;
966             scanf("%d", &tag);
967             if(tag==0){
968                 int x, y, t;
969                 scanf("%d%d%d", &x, &y, &t);
970                 update(1, x, y, t, 1, n);
971             }
972             else if(tag==1){
973                 int x, y;
974                 scanf("%d%d", &x, &y);
975                 printf("%d\n", getmx(1, x, y, 1, n));
976             }
977             else {
978                 int x, y;
979                 scanf("%d%d", &x, &y);
980                 printf("%lld\n", getsum(1, x, y, 1, n));
981             }
982         }
983     }
984 }
985
986 // #include <bitset>
987 // using std::bitset;
988 // bitset<n>b;
989 // b.any() any 1?
990 // b.none() no 1?
991 // b.count() number of 1?
992 // b.size() size?
993 // b.set() set all pos 1
994 // b.reset() set all pos 0
995 // b.flip() 1->0, 0->1
996
997 // k中浓度为ai/1000的液体配成n/1000的浓度至少需要多少升
998 // 每种溶液均为整数升
999 // 易知若有解必小于1000, dp过程用bitset加速
1000
1001 #include <bits/stdc++.h>
1002 using namespace std;
1003
1004 int n, k;
1005 bool a[1050];
1006 bitset<2050>b[2];
1007
1008 int main(){
1009     scanf("%d%d", &n, &k);
1010     for(int i=1;i<=k;i++){
1011         int x;
1012         scanf("%d", &x);
1013         a[x]=true;
1014     }
1015     b[0][1000]=1;
1016     for(int i=0;i<=1000;i++){
1017         if(i!=0&&b[i%2][1000]){
1018             printf("%d\n", i);
1019             return 0;
1020         }

```

```

1021     int now=i%2;
1022     b[now^1].reset();
1023     for(int j=0;j<=1000;j++){
1024         if(a[j]){
1025             b[now^1]|=(b[now]<<j)>>n;
1026         }
1027     }
1028 }
1029 printf("-1\n");
1030 }
1031
1032 //树状数组区间修改区间查询
1033 //sum[n]=sigma(a[i])+sigma((n+1-i)d[i]);
1034 //其中a[i]是原数组元素
1035 //d[i]是[i, n]的共同增量
1036 #include <bits/stdc++.h>
1037 using namespace std;
1038 typedef long long ll;
1039 ll a[205000];
1040 ll d[205000];
1041 ll di[205000];
1042 int n, q;
1043
1044 int lowbit(int x){
1045     return x&(-x);
1046 }
1047
1048 void update(int x, int num){
1049     ll add=1LL*num;
1050     ll addi=1LL*num*x;
1051     while(x<=n){
1052         d[x]+=add;
1053         di[x]+=addi;
1054         x+=lowbit(x);
1055     }
1056 }
1057
1058 ll query(int x){
1059     ll ans=a[x];
1060     ll tmp=x;
1061     while(tmp){
1062         ans+=d[tmp]*(x+1);
1063         ans+=di[tmp]*(-1);
1064         tmp-=lowbit(tmp);
1065     }
1066     return ans;
1067 }
1068
1069 int main()
1070 {
1071     scanf("%d", &n);
1072     for(int i=1;i<=n;i++){
1073         scanf("%lld", &a[i]);
1074         a[i]+=a[i-1];
1075     }
1076     scanf("%d", &q);
1077     while(q--){
1078         int tag;
1079         scanf("%d", &tag);
1080         if(tag==1){
1081             int x, y, num;
1082             scanf("%d%d%d", &x, &y, &num);
1083             update(x, num);
1084             update(y+1, -num);
1085         }
1086         else {
1087             int x, y;
1088             scanf("%d%d", &x, &y);
1089             printf("%lld\n", query(y)-query(x-1));
1090         }
1091     }
1092 }
1093

```