hnfms noip2023_r1

题目名称	flandre	meirin	sakuya	红楼 ~ Eastern Dream
题目类型	传统型	传统型	传统型	传统型
目录	`flandre`	`meirin`	`sakuya`	`scarlet`
可执行文件名	`flandre`	`meirin`	`sakuya`	`scarlet`
输入文件名	`flandre.in`	`meirin.in`	`sakuya.in`	`scarlet.in`
输出文件名	`flandre.out`	`meirin.out`	`sakuya.out`	`scarlet.out`
每个测试点时限	1000ms	500ms	1000ms	800ms
内存限制	128.00MB	128.00MB	256.00MB	32.00MB
子任务数目	5	20	20	20
结果比较方式	Special Judge	全文比较（忽略行末空格及文末回车）	全文比较（忽略行末空格及文末回车）	全文比较（忽略行末空格及文末回车）

编译选项 `-std=c++14 -O2`

flandre

本题开启Subtask

题目背景

二小姐的烟花 多 好 看 啊!

题目描述

芙兰朵露有 n 种烟花，每种都有两个参数：「真实效果」a 和「感觉效果」b，其中「真实效果」是一个给定不变的整数（可以为负），「感觉效果」初值等于「真实效果」。

将烟花按一定顺序燃放，先燃放的烟花会使得后面「真实效果」较差的烟花「感觉效果」更差，「真实效果」更优的「感觉效果」更优。

具体来说，**对于所有** $i < j$，即烟花 i 在 j 前燃放，则有以下三种情况：

- $a_i < a_j$ 则 $b_j \leftarrow b_j + k$
- $a_i = a_j$ 则 b_j 不变
- $a_i > a_j$ 则 $b_j \leftarrow b_j - k$

其中 k 为给定常数。

芙兰朵露想使「感觉效果」总和尽量大。也就是说，她想选一部分（可以全选或不选，不能重复选择）按一定顺序燃放，假设选了 m 个，使 $\sum_{i=1}^{m} b_i$ 最大。输出最大值及方案，有多解输出其一即可。

输入格式

第一行两个正整数 n, k。

第二行 n 个用空格隔开的整数 $a_{1\dots n}$ 依次表示 1 号到 n 号烟花的「真实效果」。

输出格式

第一行两个整数表示答案 $\max\{\sum\limits_{i=1}^{m} b_i\}$ 和选择的烟花数量。

第二行依次输出选择烟花的编号，表示按此顺序燃放烟花。

样例 #1

样例输入 #1

```
3 1
1 2 3
```

样例输出 #1

```
9 3
1 2 3
```

样例 #2

样例输入 #2

```
6 6
1 1 4 5 1 4
```

样例输出 #2

```
82 6
2 1 5 6 3 4
```

提示

样例解释

样例#1

最后的「感觉效果」依次为 $1, 3, 5$。

样例#2&#3

没有解释。

数据范围

Subtask	特殊限制	分值
1	$n \le 10$	20
2	$a_i \ge 0$	10
3	$n \le 1000$，数据随机	20
4	数据随机	20
5	无	30

对于 100% 的数据：

- $1 \le n \le 10^6$
- $0 \le k \le 10^6$
- $-10^6 \le a_i \le 10^6$

本题第 4 个样例只给出 $\max\{\sum\limits_{i=1}^{m} b_i\}$ 的值，所以第 4 个样例的输出只有一个数。

题解

题意解释

题目给出一个序列，然后让你选出它的任意子序列的一个排列，使得正序对数减逆序对数的差乘 k 加所有数的和最大。

知识点提炼

排序，贪心，数学结论。

核心解题思路

子任务 1

直接爆搜模拟即可。

子任务 2

不难发现，答案序列一定是有序的，这样既可以最大化正序对数，也可以使逆序对数为 0。

而 $a_i > 0$，所以我们尽量全部选上每个数字，这样一定是不劣的，把所有数排序后即可。

子任务 3,4

如果猜结论，可以猜到答案是原序列排序后的一段后缀，而数据随机，我们可以认为 a_i 互不相同。

此时结论很好证明，证明如下：

考虑反证。

设 $b_i = a_i + p_i k$（待定系数），A 的长度为 m。

假设答案不为后缀，而是由两个区间 $[l_1, r_1], [l_2, m]$ 组成的。

那么可以把区间 $[l_1, r_1]$ 向右平移，直到与 $[l_2, m]$ 合并更优。

这是因为，对于前后的一对 b_i 与 b'_i，显然 $a'_i > a_i$，而 $p'_i = p_i$。

所以这个简单版就证完了。

子任务 5

考虑拓展结论到任意序列，证明如下：

现在定义一些东西，记 $f(\{A\}_n) = \sum\limits_{i=1}^{n} A_i + A\text{排序后的顺序对} \times k$，不难发现，答案就是 f 的最大值。

再记 $val(A, x) = F(A + x) - F(A)$，$A + x$ 是指将 x 插入序列 A 得到的序列。

你的最优方案 A 一定要满足 $val(A - lastnum, lastnum) > 0$，否则我可以不加入最后一个数。

然后假如原序列排序后是这样的：$[\ldots x, y, y, y, y, z \ldots]$，而最优方案是 $[x, y, z \ldots]$。

假如你最后加入了 x，则 $val([y, z \ldots], x) > 0$，而发现在 $[x, y, z \ldots]$ 中加入 y 后，与加入 x 后的顺序对的增加量是相同的，而 $y > x$ 则 $val([x, y, z \ldots], y) > val([y, z] \ldots, x) > 0$ 所以此时加上 y 更优。

而且继续加 y 的 val 是一样的，所以你还可以继续加 y，你就可以把不连续段合并，所以答案必定是一个排序后的后缀。

现在做法就显然了，把 a 的每个后缀的答案像算后缀和一样都算出来，取最大方案即可。

代码如下。

```
#include<bits/stdc++.h>
#define int long long
```

```
using namespace std;
int n,k;
struct node{
    int pos,val;
}a[1000001];
bool cmp(node x,node y){
    return x.val<y.val;
}
map<int,int> ma;
int ans=0,p=n+1;
int now=0;
signed main(){
    cin>>n>>k;p=n+1;
    for(int i=1;i<=n;i++){
        cin>>a[i].val;
        a[i].pos=i;
    }
    sort(a+1,a+1+n,cmp);
    for(int i=n;i;i--){
        now+=a[i].val+k*(n-i-ma[a[i].val]);
        ma[a[i].val]++;
        if(now>ans){
            ans=now;
            p=i;
        }
    }
    cout<<ans<<' '<<n-p+1<<'\n';
    for(int i=p;i<=n;i++){
        cout<<a[i].pos<<' ';
    }
}
```

本题的结论完整证明其实不容易在考场证明，但可以通过大胆猜测来得到结论，从而通过此题。

易错点

要开 `long long`，答案可以不选任何数。

meirin

题目背景

红美铃又在工作的时候摸鱼了。

她经常在帕秋莉的图书馆借书。作为妖怪，她有着非常强大的学习能力。所以她也学习到了一些 OI 相关的知识！

她偶然看到了 luoguP5686。

题意是给出两个序列 a,b ，求 $\sum\limits_{l=1}^n\sum\limits_{r=l}^n(\sum\limits_{i=l}^r a_i)\times(\sum\limits_{i=l}^r b_i) \bmod 1000000007$。

她当然做出来了。

刚好她才学过线段树，会区间加法。

于是她就把区间加法操作搬到了这题上。

但是此时十六夜咲夜发现她了在摸鱼，于是赶紧放下题目留给了你。

题目描述

给出两个长度为 n 的序列 a,b

q 次操作，每次格式如下

给出 `l r k` 将序列 b 区间 $[l,r]$ 的值每个加上 k。

每次操作结束后，查询：

$$\sum_{l=1}^n\sum_{r=l}^n(\sum_{i=l}^r a_i)\times(\sum_{i=l}^r b_i) \bmod 1000000007$$

输入格式

第一行输入两个整数 n,q。

然后一行 n 个整数，表示序列 a。

一行 n 个整数，表示序列 b。

最后 q 行，每行三个整数 l,r,k 表示一次修改。

输出格式

q 行，表示当前序列 $\sum\limits_{l=1}^n\sum\limits_{r=l}^n(\sum\limits_{i=l}^r a_i)\times(\sum\limits_{i=l}^r b_i) \bmod 1000000007$ 的值。

样例 #1

样例输入 #1

```
6 6
1 1 4 5 1 4
1 9 1 9 8 1
1 6 0
1 1 4
1 5 4
4 6 2
1 6 -3
1 6 -20
```

样例输出 #1

```
2997
3189
5145
5731
4072
999993019
```

提示

测试点编号	特殊限制	分值
$1 \sim 4$	$n \leq 10^2, q \leq 10^2$	20
$5 \sim 8$	$n \leq 10^5, q \leq 5$	20
$9 \sim 12$	$l = r$	20
$13 \sim 16$	$a_i = 1$	20
$17 \sim 20$	无特殊限制	20

对于所有的数据，$|a_i, b_i, k| \leq 10^9, 1 \leq l \leq r \leq n, n \leq 5 \times 10^5, q \leq 10^6$

题解

题意解释

题意很清楚，区间加法操作，求 $\sum\limits_{l=1}^{n}\sum\limits_{r=l}^{n}(\sum\limits_{i=l}^{r}a_i) \times (\sum\limits_{i=l}^{r}b_i) \bmod 1000000007$ 的值。

知识点提炼

数学，前缀和。

子任务 1

直接暴力模拟计算这个式子即可。

子任务 2

考虑用 a,b 的前缀和 sa,sb 来简化式子。

$$\sum_{l=1}^{n}\sum_{r=l}^{n}(\sum_{i=l}^{r}a_i)\times(\sum_{i=l}^{r}b_i)$$

$$=\sum_{l=1}^{n}\sum_{r=l}^{n}(sa_r-sa_{l-1})\times(sb_r-sb_{l-1})$$

展开后因式分解得

$$=(n+1)\sum_{i=1}^{n}sa_isb_i-\sum_{i=1}^{n}sa_i\sum_{i=1}^{n}sb_i$$

子任务 3,5

先把式子换一种写法。

原式等于 $\sum_{l=1}^{n}\sum_{r=l}^{n}\sum_{i=l}^{r}(b_i\times(\sum_{j=l}^{r}a_j))$

显然可以待定系数一下：

设答案为 $\sum_{i=1}^{n}b_iq_i$ 。

再设 $q_i=\sum_{j=1}^{n}p_{i,j}a_j$

由于修改是与 b 有关而与 a 无关的，所以 p,q 都不会修改。

看一下这样处理有什么好处。

对于一个区间 $[l,r]$ 增加 k，设它对原答案就会增加 $\sum_{i=l}^{r}kq_i=k\sum_{i=l}^{r}q_i$ 。

那么只要求出 q 就好了。

而求 q 是与 p 有关的，考虑一下 $p_{i,j}$ 的含义。

观察式子可以发现，它指的是既包含了 i，并且包含了 j 的区间的个数。（这里可以手动模拟思考一下）

那么当 $i\le j$ 时， $q_{i,j}=i\times(n-j+1)$

否则为 $j\times(n-i+1)$。

所以 $p_i=\sum_{j=1}^{i-1}j(n-i+1)a_j+\sum_{j=i}^{n}i(n-j+1)a_j$

但现在去求 p 依然是 n^2 的。

所以可以观察 p_i 与 p_{i-1} 的关系。

显然先把一个 p 分为两个 $\sum$ 加起来，设它们是 $p_i=x_i+y_i$。

观察一下，$x_i=x_{i-1}-\sum_{j=1}^{i-1}ja_j$，

$y_i=y_{i-1}+\sum\limits_{j=i}^{n}(n-j+1)a_j$。

所以预处理一下 ia_i 的前缀和与 $(n-i+1)a_i$的后缀和就做完了，时间复杂度很优秀，是 $O(n+m)$ 的。

子任务 4

如果你不会处理 $q_i=\sum\limits_{j=1}^{n}p_{i,j}a_j$ 的话，就可以在 $a_i=1$ 的情况下做出来。

正解代码（代码仅供参考，为了保证可读性，没有加任何优化，不一定能过）：

```
#include<bits/stdc++.h>
#define int long long
#define mod 1000000007
using namespace std;
int n;
int sum1[1000005],sum2[1000005];
int a[1000005],b[1000005],ans;
int ask1(int l,int r){
    if(r<l) return 0;
    return (sum1[r]-sum1[l-1]+mod)%mod;
}
int ask2(int l,int r){
    if(r<l) return 0;
    return (sum2[r]-sum2[l-1]+mod)%mod;
}
int sum[1000005];
int q;
signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    cin>>n>>q;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        sum1[i]=i*a[i]%mod;
        sum2[i]=(n-i+1)*a[i]%mod;
    }
    for(int i=1;i<=n;i++) cin>>b[i];
    for(int i=1;i<=n;i++){
        sum1[i]=sum1[i-1]+sum1[i],sum1[i]%=mod;
        sum2[i]=sum2[i-1]+sum2[i],sum2[i]%=mod;
    }
    int S=0;
    for(int pos=1;pos<=n;pos++){
        S+=ask2(pos,n);
        S-=ask1(1,pos-1);
        S=(S+mod)%mod;
        sum[pos]=S;
        ans=(ans+S*b[pos]%mod)%mod;
    }
    for(int i=1;i<=n;i++){
        sum[i]=sum[i-1]+sum[i];
        sum[i]%=mod;
    }
```

```
    while(q--){
        int l,r,k;
        cin>>l>>r>>k;
        ans+=((sum[r]-sum[l-1])*k%mod+mod);
        ans=(ans%mod+mod)%mod;
        cout<<ans<<'\n';
    }
}
```

易错点

如果不保证你的常数，不要 `define int long long`，这样常数大，然后尽量减少取模操作。

sakuya

题目描述

红魔馆的内部形态是一棵树，总共有 n 个房间，其中它们由 $n-1$ 个走廊连接。

每条走廊都有一个难走程度 w 。

每一天，十六夜咲夜都会打扫房间。

她在打扫房间的时候会使用能力，暂停时间。所以打扫一个房间不要任何时间。然而进过一个走廊是要花费时间的，所话花费的时间就是这个走廊的难走程度。

现在给出 m 个特殊房间，把这些房间记为集合 A 。现在十六夜咲夜想随机的从一个房间开始打扫，然后随机进入另一个 A 中没有被打扫的房间去打扫它，直到把这 m 个房间都打扫干净。

现在，她想知道她把这些房间打扫干净的期望时间对 998244353 取模的值。

换句话说，设 $d(x,y)$ 表示从树上点 x 到 y 的边权和。

将 A 随机打乱得到序列 a 。

求 $\sum\limits_{i=2}^{m} d(a_{i-1},a_i) \mod 998244353$ 。的期望值

但是每天早上，帕秋莉会在某一个房间 x 使用魔法，魔法有一个强度 k 。这会使得所有与 x 相连的走廊的难走程度增加 k。（十六夜咲夜：谢谢你啊。）

而你需要回答每天使用魔法后的期望值。

输入格式

第一行两个整数 n,m。

接下来 n 行，每行三个整数 u,v,w，表示 u 到 v 有一条难走程度为 w 的边。

接着一行 m 个整数，表示 m 个特殊房间。

然后输入 q，表示询问个数。

q 行，每行两个整数 x,k，表示在点 x 使用强度为 k 的魔法。

输出格式

q 行，表示每次修改后的期望值。

样例 #1

样例输入 #1

```
4 3
1 2 1
1 3 2
3 4 3
2 3 4
3
1 0
4 1
1 5
```

样例输出 #1

```
8
332748127
665496258
```

提示

测试点编号	特殊限制	分值
$1 \sim 4$	$n, q \leq 10$	20
$5 \sim 6$	$n, q \leq 100$	10
$7 \sim 10$	$q = 1$	20
$11 \sim 12$	$m = 3$	10
$13 \sim 14$	修改操作只在叶子上	10
$15 \sim 20$	无	30

对于 100% 的数据：

$1 \leq n \leq 5 \times 10^5, m \leq n, 1 \leq q \leq 5 \times 10^5$。

$1 \leq w, k \leq 10^9$ 。

对于样例中的第一个询问，a 有 `2 3 4`,`2 4 3`,`3 2 4`,`3 4 2`,`4 2 3`,`4 3 2`这 6 种可能，它们的值分别为 $6, 9, 9, 9, 9, 6$，所以期望为 $\dfrac{6+9+9+9+9+6}{6} = 8$

题解

题意解释

题目给出一棵树，树上有一些关键点，它们构成排列 A，你在随机化打乱 A 后，求出 $\sum\limits_{i=2}^{n} d(A_{i-1}, A_i)$ 的期望。

并且还会给出 m 次修改，每次增加一个点周围所有边的边权。

知识点提炼

期望，数学，树形dp

子任务1

首先不难发现，这个期望是假的，其实是平均数

模拟每一个排列，求出权值再求平均值即可。

子任务2

既然发现是平均数，那么可以把式子转换成求 $\frac{\sum_{i,j} d(i,j) \times num_{i,j}}{m!}$，其中 $num_{i,j}$ 为 i,j 在序列中相邻出现的次数。

由于是所有排列，不难发现，对于任意 i,j，有 $num_{i,j} = 2(m-1)!$ 。

于是问题变成了求 $\sum_{i,j} d(i,j)$ 如果暴力dp求出每个$d(i,j)$ 的值即可通过这档分。

子任务3

把上述的暴力dp换成换根dp即可。

子任务 4

由于 m 小，式子只有几项，于是可以数据结构算两两点的距离即可。

子任务 5

先用子任务3的做法求出初始答案。

每次只修改一条边于是只用求出这条边被经过的次数再乘上 k 即可。

子任务6

有了子任务5，可以不难发现正解了。

记连接 x,y 条边在 $\sum_{i,j} d(a_i,a_j)$ 这个式子中被计过的边数为 $num_{x,y}$，那么一次操作对这个式子的修改是增加了 $k \sum_{E(x,u)} num_{x,u}$ ，显然，预处理出 num 后可以 $O(1)$ 计算。

那么树形dp处理的部分就是求出 num，我们这样考虑一条边，一条边可以把树分为两个部分，那么就是这两个部分的特殊点的个数乘积就是 num 的值了，树形dp预处理子树中特殊点的个数即可。

代码如下

```
#include<bits/stdc++.h>
#define int long long
#define mod 998244353
using namespace std;
int quickpow(int a,int b){
    int ret=1;
    while(b){
        if(b&1) ret=ret*a%mod;
        a=a*a%mod;
        b>>=1;
```

```
    }
    return ret;
}
int inv(int x){
    return quickpow(x,mod-2);
}
int n,m,q;
struct edge{
    int from,to,val;
}e[2000001<<1];int head[2000001],size;
void addedge(int x,int y,int z){
    e[++size].to=y,e[size].val=z;
    e[size].from=head[x],head[x]=size;
}
int dep[2000001],p[2000001],f[2000001];
bool vis[2000001];
int tot;
void dfs1(int now,int fa){
    if(vis[now]) p[now]++;
    dep[now]=dep[fa]+1;
    for(int i=head[now];i;i=e[i].from){
        int u=e[i].to;
        if(u==fa) continue;
        dfs1(u,now);
        p[now]+=p[u];
    }
}
int get(int x,int y){
    if(dep[x]>dep[y]) return p[x]*(m-p[x]);
    return p[y]*(m-p[y]);
}
int ans;
void dfs2(int now,int fa){
    for(int i=head[now];i;i=e[i].from){
        int u=e[i].to;
        f[now]+=get(now,u);
        f[now]%=mod;
        if(u==fa) continue;
        dfs2(u,now);
        ans+=e[i].val*get(now,u);
        ans%=mod;
    }
}
signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m;
    int times=inv(m)*2%mod;
    for(int i=1;i<n;i++){
        int x,y,z;
        cin>>x>>y>>z;
        addedge(x,y,z);
        addedge(y,x,z);
    }
    for(int i=1;i<=m;i++){
```

```
        int x;cin>>x;
        vis[x]=1;
    }
    dfs1(1,1);
    dfs2(1,1);
    cin>>q;
    while(q--){
        int x,k;cin>>x>>k;
        ans+=f[x]*k%mod;ans%=mod;
        cout<<ans*times%mod<<'\n';
    }
}
```

红楼 ~ Eastern Dream

题目描述

给出一个长度为 n 的序列 a，有 m 次操作，格式如下：

`1 x y k` 对于所有满足 $(i-1) \bmod x \le y$ 的 i，将 a_i 的值加 k。

`2 l r` 求 $\sum\limits_{i=l}^{r} a_i$ 。

数组下标从 1 开始。

输入格式

第一行两个正整数 n, m。

接下来一行 n 个整数表示 a_i 的值。

然后 m 行，每行表示一个操作。

输出格式

对于所有的 2 操作，输出答案。

样例 #1

样例输入 #1

```
6 7
1 1 4 5 1 4
2 1 5
1 1 4 5
2 1 5
1 3 2 1
2 4 6
1 4 2 2
2 1 6
```

样例输出 #1

```
12
37
28
62
```

提示

测试点编号	特殊限制	分值
$1 \sim 4$	$1 \le n, m \le 10^3$	20
$5 \sim 8$	$x \le 100$	20
$9 \sim 12$	$l = r$	20
$13 \sim 16$	$x \ge \frac{n}{2}$	20
$17 \sim 20$	无特殊限制	20

对于 100% 的数据，$1 \le n, m \le 2 \times 10^5, 1 \le x \le n, 1 \le a_i, k \le 10^9$。

注意本题空间限制

题目大意

题意很清楚，是个显然的数据结构。

知识点提炼

根号分治，根号平衡，分块。

子任务1

暴力模拟即可。

子任务2

发现到每次修改操作其实本质是讲序列分为 $\frac{n}{x}$ 段，每段进行一次前缀上的加法。

而此时 x 很小，于是我们对于每个 x，记录其循环节，然后每次修改只维护循环节与循环节的前缀和。

询问时遍历每个 x，计算询问区间内有多少完整的循环节与散的循环节的区间和即可。

时间复杂度 $O((n+m)x)$。

子任务4

x 相当的大序列只被分成了两段。，于是可以每次暴力用线段树进行修改，然后线段树查询。

时间复杂度 $O((n+m)\log n)$

子任务3

有了刚刚的思考，我们便可以自然想到根号分治。

对于 $x \le \sqrt{n}$ 跑子任务2

否则对于小于 $\sqrt{n}$ 个段，每个段都暴力线段树修改，这样可以做到修改 $\sqrt{n}\log n$，查询 $\log n$。

显然要根号分治，如果只是为了解决子任务3，可以做区间修改 $O(1)$，单点查询 $O(\sqrt{n})$ 的分块，这是简单的，不多赘述，于是可以通过这档分。

子任务5

我们要做到**区间**修改 $O(1)$，**区间**查询 $O(\sqrt{n})$ 的分块。

你要 $O(1)$ 区间加法，那么必然是差分维护的。

那么查询区间 $[l,r]$ 的和就是 $\sum\limits_{i=l}^{r}\sum\limits_{j=1}^{i}c_j$ （c 是差分数组），这是由差分的定义得到的，推一下式子：

$$\sum_{i=l}^{r}\sum_{j=1}^{i}c_j=\sum_{i=1}^{l-1}c_i\times(r-l+1)+\sum_{i=l}^{r}c_i(r-i+1)=(r-l+1)\sum_{i=1}^{l-1}c_i+\sum_{i=l}^{r}c_i(r-i+1)$$

左边那个式子很方便分块查询，主要是右边，它的系数是一个类似这样变化的东西：

我们要处理那段下降的直线与 c_i 的乘积，也就是 $\sum\limits_{i=l}^{r}(r-i+1)c_i$ 的值。

利用分块，我们干这样的一件事：

每一段蓝色三角形的底长为 $\sqrt{n}$，每一块分别维护出 $\sum\limits_{i=L}^{R}c_i(R-i+1)$ ，其中 L,R 是这个块的左右端点。

那么对于一个整块的答案就是 $(r-R)\sum\limits_{i=L}^{R}c_i+\sum\limits_{i=L}^{R}c_i(R-i+1)$ 两个 $\sum$ 都可以在修改时 $O(1)$ 维护，那么整个部分是 $O(1)$ 的，而零散块可以直接暴力套用刚刚的式子计算，查询时间复杂度为 $O(\sqrt{n})$。

综上，算法总复杂度为 $O(n\sqrt{n})$ ，做完了。

代码如下：

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
int n,m,a[500001];
int in[500001],size;
int sum[1001],ssum[1001],le[1001],re[1001];
int t[500001];
void add(int l,int r,int k){
    t[l]+=k;sum[in[l]]+=k;
    ssum[in[l]]+=(re[in[l]]-l+1)*k;
    if(r==n) return;
    t[r+1]-=k;sum[in[r+1]]-=k;
    ssum[in[r+1]]-=(re[in[r+1]]-r)*k;
}
long long ask(int l,int r){
    long long ret=0;
    for(int i=1;i<in[l];i++){
        ret+=sum[i]*(r-l+1);
    }
    for(int i=le[in[l]];i<l;i++){
        ret+=t[i]*(r-l+1);
    }
    if(in[l]==in[r]){
        for(int i=l;i<=r;i++){
```

```
            ret+=t[i]*(r-i+1);
        }return ret;
    }
    for(int i=l;i<=re[in[l]];i++){
        ret+=t[i]*(r-i+1);
    }
    for(int i=in[l]+1;i<in[r];i++){
        ret+=ssum[i]+sum[i]*(r-le[i+1]+1);
    }
    for(int i=le[in[r]];i<=r;i++){
        ret+=t[i]*(r-i+1);
    }
    return ret;
}
int d[501],ds[1000001];
int askd(int x,int y){
    if(y==-1) return 0;
    return (y/x)*d[x]+ds[x*1000+y%x];
}
signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        a[i]+=a[i-1];
    }
    size=sqrt(n)/2;
    for(int i=1;i<=n;i++){
        in[i]=((i-1)/size)+1;
        if(!le[in[i]]) le[in[i]]=i;
        re[in[i]]=i;
    }
    int tot=0;
    while(m--){
        int op,l,r,k,x,y;
        cin>>op;
        if(op==1){
            cin>>x>>y>>k;
            y=min(x-1,y);
            if(x<=size){
                d[x]+=k*(y+1);
                int s=0;
                int X=x*1000;
                for(int i=0;i<=y;i++){
                    s+=k;
                    ds[X+i]+=s;
                }
                for(int i=y+1;i<x;i++){
                    ds[X+i]+=s;
                }
            }
            else{
                l=1;
                while(l<=n){
```

```
                    add(l,l+y,k);
                    l+=x;
                }
            }
        }
        else{
            cin>>l>>r;
            tot++;
            int ans=a[r]-a[l-1];
            for(int i=1;i<=size;i++){
                ans+=askd(i,r-1)-askd(i,l-2);
            }
            ans+=ask(l,r);
            cout<<ans<<'\n';
        }
    }
}
```

易错点

要卡常。