T1-锁链战车

因为不能出现环，所以将森林中每棵树的直径连起来就是最长的链。

T2-刺探军情

令 V 表示值域的上限， $g(l,r)$ 表示 $[l,r]$ 的区间 $\gcd$ 。

对于 $g(1,r)$，$g(2,r)$，$\ldots g(r,r)$ 的取值一定是单调不降的，且最多有 $logV$ 个不同的取值。

算法一： $O(n\times logn\times log^2V)$

用 ST 表预处理出 $g(l,r)$ 。然后枚举右端点 r ，对于每个 r 二分出来 $logV$ 个不同的 g 取值的最靠左的点，计算并取 max 即可。

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int N = 1e6 + 5;
int n, k, h[N], g[N][21], lg[N];
ll sum[N], ans;
inline int gcd(int l, int r) {
    int k = lg[r - l + 1];
    return __gcd(g[l][k], g[r - (1 << k) + 1][k]);
}
inline ll solve(int r) {
    int l = 1;
    ll res = 0;
    while (l <= r - k + 1) {
        const int glr = gcd(l, r);
        res = max(res, glr * (sum[r] - sum[l - 1]));
        if (glr == h[r])
            break;
        int L = l + 1, R = r, pos = r;
        while (L <= R) {
            int mid = (L + R) >> 1;
            if (gcd(mid, r) > glr)
                pos = mid, R = mid - 1;
            else
                L = mid + 1;
        }
        l = pos;
    }
    return res;
}
signed main() {
    cin.tie(nullptr)->sync_with_stdio(false);
    cin >> n >> k;
```

```
    for (int i = 1; i <= n; i++)
        cin >> h[i];
    for (int i = 1; i <= n; i++)
        sum[i] = sum[i - 1] + h[i];
    for (int i = 2; i <= n; i++)
        lg[i] = lg[i >> 1] + 1;
    for (int i = 1; i <= n; i++)
        g[i][0] = h[i];
    for (int j = 1; j <= lg[n]; j++)
        for (int i = 1; i + (1 << j) - 1 <= n; i++)
            g[i][j] = __gcd(g[i][j - 1], g[i + (1 << (j - 1))][j - 1]);
    for (int i = k; i <= n; i++)
        ans = max(ans, solve(i));
    cout << ans << endl;
    return 0;
}
```

算法二： $O(n \times log^2 V)$

发现每次 $r+1$ 时只需要对这 $\log V$ 个不同的 g 再分别对 h_{r+1} 取 $\gcd$ 即可，去重后依旧是 $\log V$ 的，所以直接维护就好了，复杂度 $O(n \log^2 V)$。

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int N = 1e6 + 5;
int n, k, h[N], g[21], pos[21], tot;
ll sum[N], ans;
signed main() {
    cin.tie(nullptr)->sync_with_stdio(false);
    cin >> n >> k;
    for (int i = 1; i <= n; i++)
        cin >> h[i];
    for (int i = 1; i <= n; i++)
        sum[i] = sum[i - 1] + h[i];
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= tot; j++)
            g[j] = __gcd(g[j], h[i]);
        if (g[tot] != h[i])
            g[++tot] = h[i], pos[tot] = i;
        int ntot = 0;
        for (int j = 1; j <= tot; j++)
            if (g[j] != g[j - 1])
                g[++ntot] = g[j], pos[ntot] = pos[j];
        tot = ntot;
        for (int j = 1; j <= tot; j++)
            if (i - pos[j] + 1 >= k)
                ans = max(ans, g[j] * (sum[i] - sum[pos[j] - 1]));
    }
    cout << ans << endl;
```

```
    return 0;
}
```

T3-一击制敌

Subtask 1

暴力穷举区间，枚举众数判断即可。复杂度 $\mathcal{O}(n^3)$。

Subtask 2

暴力穷举区间，区间元素个数用一个桶维护起来，区间 $(i,j+1)$ 的各个元素的个数可以由区间 (i,j) 递推得。但由于 a_i 得范围达到了 10^9，需要离散化一下。复杂度 $\mathcal{O}(n^2)$。

Subtask 3

考虑到只有 1，2 两种数字，所以区间不合法只有一种情况：区间内 1，2 个数相等。考虑将 1 的值设成 1，2 的值设成 -1，所以不合法的区间的区间和为 0，可以记录一个前缀桶来维护。复杂度 $\mathcal{O}(n)$。

Subtask 4

受 Subtask 3 的启发，穷举每个种类的数，分别计算贡献。

$a_i \le 10^9$ ，离散化一下就可以了。考虑当前计算的是 x 的贡献，记 S_i 为前 i 个数中 x 出现的次数，则 $[l+1,r]$ 区间合法则需

$$2(S_r - S_l) > r - l$$

移一下项就变为

$$2S_r - r > 2S_l - l$$

这样问题就变成了：对于每一个 r，$0 \sim r-1$ 中由多少个 l 使 $2S_l - l < 2S_r - r$，是一个二维偏序问题，可以通过树状数组维护，复杂度是 $\mathcal{O}(kn\log n)$ 的，其中 k 为不同数字的个数。但是这样还是只能通过前三个 Subtask，需要优化。

设 $B_i = 2S_i - i$，对于每一个 x，考虑 B_i 的变化情况。举个例子：

	A	B	C	D	E	F	G	H	I	J	K	L
1	下标	0	1	2	3	4	5	6	7	8	9	10
2	原序列	0	2	2	1	2	2	1	2	2	1	2
3	x=1时的Si	0	0	0	1	1	1	2	2	2	3	3
4	x=1时的Bi	0	-1	-2	-1	-2	-3	-2	-3	-4	-3	-4

可以发现如果有 m 个 x，那么 B_i 可以被分成 $m+1$ 个区间，每个区间都是一个公差为 -1 的等差数列。考虑到所有的 m 的和为 n，我们其实只需要快速处理每一段里的数。

假设这个等差数列为 $s, s-1, \cdots, e+1, e$，用一个数组 C_i 来记录每一个数的个数，那么就是 $[e,s]$ 区间的每个 C_i 都加 1。

设 D_i 表示 C_i 的前缀和，即 $D_i=\sum\limits_{j=1}^{i}C_j$，对于每个 B_i，贡献即为 D_{B_i-1}，所以对于 $[e,s]$ 这个区间内的所有数，总贡献即为 $\sum\limits_{i=e-1}^{s-1}D_i$，这又变成了一个区间和问题，我们再维护一个数组 E_i 表示 C_i 前缀和的前缀和，即为 D_i 数组的前缀和，即 $E_i=\sum\limits_{j=1}^{i}D_j$，这样总贡献就变成了

$$\sum_{i=e-1}^{s-1}D_i=E_{s-1}-E_{e-2}$$

至此问题就变成了一个**区间修改，求二阶前缀和**的问题。这就有很多解法了，这里只提供一个树状数组的解法。

考虑到区间修改可以通过**单点修改，求前缀和**来解决，问题就可已转化为 **单点修改，求三阶前缀和。**

分治做法：$O(n\times log^2n)$

> 提供一种考场上想出来的 $n\log^2n$ 分治做法，应该比正解好实现许多。
>
> ---
>
> 首先由于是序列计数问题，自然要往单调性方面去想。可仔细思考就会发现，众数这东西的性质好像不那么美妙，区间扩展时并不具有很好的性质，不能考虑枚举左端点然后向右扩展。
>
> 既然区间扩展不行，就考虑区间拼接。
>
> 我们先考虑两个区间拼起来是合法的，需要满足什么条件？我们用 L_x,R_x 分别表示 L,R 区间喜欢 x 菜肴的人数，那么 L,R 拼接起来合法，一定是存在一个 x，使得 $2\times(L_x+R_x)>len_L+len_R$，不难发现不等式拆开就是一个一维偏序，将序列离散化后两组区间的拼接可以 $O(n)$ 解决。
>
> 可现在还有一个问题，上面偏序的前提是 x 确定，可我们怎么知道 x 是什么呢？不难发现，对于两个确定的区间拼接，x 必须是其中一方的众数，可这并不能帮助我们把许多区间更快拼接。
>
> 可让我们一看题目,你就会发现题目还有一个很强的条件：获得一半以上的人支持!
>
> 我们定义一个序列有绝对众数，当且仅当一个序列的众数个数大于区间长度除以二。这时我们可以将上面的结论改一改：对于两个确定的区间拼接，x 必须是其中一方的绝对众数。
>
> 考虑序列分治，每次递归处理子区间的问题，现在我们只考虑跨过 mid 的区间。考虑分治带给我们什么好处：不难发现，对于左半部分，这些区间右端点固定，对于右边部分，这些区间左端点固定。
>
> 那这又有什么用呢？不难发现，一端端点固定的一组序列，不同的绝对众数的个数是 $\log n$ 级别的。这也很好证明，因为若 $[L,R]$ 这段区间绝对众数为 a ,那么往左右扩展时要求众数改变并且是绝对众数，每次几乎都需要成倍级别增长，结论得证。
>
> 那这时思路就明确了，先序列分治，然后对于每层来说，先求出可能的绝对众数作为 x，然后枚举绝对众数用桶做一维偏序。
>
> 并且这样做不会算重，因为拼接后的一个序列一定只会被一个绝对众数判断为合法。

T4-仓皇逃跑

$Subtask\ 1$

首先合法路径上至少有两条边颜色不同，这不太好统计，我们可以反过来想：

合法方案数 = 总方案数 - 不合法的方案数

总方案数显然是 k^{n-1}, 考虑计算不合法方案数。

先考虑 $m=1$ 的情况：

显然，不合法方案的这一条路径上的颜色都是相同的，我们就想到像缩点一样，把这一条路径的边缩成一条边，这样不合法方案数就好算了，设这一条路径包含 c 条边，则不合法方案数为 k^{n-c}。

$Subtask\ 1+3$

考虑树上差分，知道每条边被算了几次，没算的边拎出来，记为 cnt。对于一条路径 u,v，合法情况就是 $k^{dis(u,v)}-k$，全部相乘，最后再乘上 k^{cnt}。

$Subtask4$

直接暴搜每条边，统计答案。

满分做法：

还是接着最早的思路想，来考虑不合法的方案。

如果 $m\geq 2$，这样算会重复计算多条路径同时不合法的方案，我们发现 $n\leq 60,m\leq 15$，于是可以直接暴力枚举容斥。

具体来说，就是枚举不合法路径的集合 S，对于每条路径，把它所有边合并到某一条边上（如果路径有相交，当然是合并成一条边了）。然后统计合并后的边数 cnt，当 $|S|$ 为偶数，让答案加上 cnt， 当 $|S|$ 为奇数，让答案减去 cnt，枚举路径集合可以直接状态压缩，合并可以用并查集实现，时间复杂度为 $O(nm2^m)$ 。