A. 乘积

注意到 b 数组没有被改动，有用的元素只会是最小值或最大值。

a 数组可能会被删掉最小值或最大值，可能有用的元素除了最小值和最大值之外还有次小值和次大值。

数据组数不多，因此可以将 a 数组的大小缩小到 $\min(N,4)$，将 b 数组的大小缩到 $\min(M,2)$，然后枚举即可。

注意答案可能超过 2^{32}，而且 Chen 博士可能选择不使用魔法。

B. 排序

通过模拟样例，很容易观察到 k 次迭代后新数组的第 i 个元素来自原数组的区间 $[1,\min(i+k,n)]$ 中，所以我们可以设计一个算法，对于任意的参数 k，在 $\mathcal{O}(n\log n)$ 的时间内将原始数组转换为新数组。

这里给出两种解法。一种是基于上述算法对答案进行二分，因为每次迭代将使得数组在字典序中的排名变得更靠前。不过，最小可能的 k 是很容易确定的，因为不难将排序前后两个数组中元素一一匹配，所以另一个解法是找到最小可能的 k，然后检查这样的转换是否能实现。前者的时间复杂度是 $\mathcal{O}(n\log^2 n)$，而后者的时间复杂度是 $\mathcal{O}(n\log n)$。

C. 生成器

对于固定的 M，显然存在一个最小的正整数 K_M，使得任何 N $(N\geq K_M)$ 个可能重复的整数里一定存在一个大小为 M 的子集满足它的元素之和也是 M 的倍数。举例来说，只要模 M 意义下的某一种数字出现至少 M 次，就必然能找到一组解，这意味着 $K\leq(M-1)^2+1$。事实上，可以通过很多方法证明 $K=2M-1$，这个也被叫做 Erdös-Ginzburg-Ziv 定理。下面给出一个利用初等技巧进行反证的解法。

若 $M=1$，显然成立。

若 $M=p\cdot q$ $(p,q>1)$，则可以从 $2M-1$ 个整数里依次找出 $2q-1$ 组大小为 p 的集合满足元素之和分别是 p 的倍数，然后再从这 $2q-1$ 组里找出 q 个集合，使得这 q 个集合的元素之和是 M 的倍数。因此只需要证明当 M 是质数时能够找到即可。

若 M 是质数，不妨假设所有从 $2M-1$ 个整数里选出 M 个整数的方案都满足其元素之和不是 M 的倍数，也即元素之和与 M 互质，然后尝试推出矛盾。令这 $2M-1$ 个数是 $a_1,a_2,\cdots,a_{2M-1}$，以及 $J=\{1,2,\cdots,2M-1\}$，考虑如下式子

$$S=\sum_{I\subset J,|I|=M}\left(\sum_{i\in I}a_i\right)^{M-1}$$

基于假设，我们有 $\gcd(\sum_{i\in I}a_i,M)=1$。结合费马小定理，可知 $\left(\sum_{i\in I}a_i\right)^{M-1}\equiv 1\pmod M$，因此 $S\equiv\sum_{I\subset J,|I|=M}1\equiv\binom{2M-1}{M}\pmod M$。结合卢卡斯定理，可知 $S\equiv\binom{1}{1}\binom{M-1}{0}\equiv 1\pmod M$。

另一方面，尝试将 S 表示成许多关于 a_i 的幂次之和，可以得知 $\prod_{k=1}^{t}a_{p_k}^{e_k}$ $(1\leq t\leq M-1,\sum_{k=1}^{t}e_k=M-1)$ 在其中的系数为 $\binom{2M-1-t}{M-1}\frac{(M-1)!}{\prod_{k=1}^{t}e_k!}$。再次结合卢卡斯定理可

知 $\binom{2M-1-t}{M-1} \equiv \binom{1}{0}\binom{M-1-t}{M-1} \equiv 0 \pmod{M}$，因而会得出 $S \equiv 0 \pmod{M}$，产生矛盾。因此假设不成立，即存在至少一个方案满足其元素之和是 M 的倍数。

动态规划可以直接找出一组解，做法是定义 $f(i,j,k)$ 表示前 i 个数里选出 j 个数使得它们之和在模 M 意义下是否可以为 k，初始只有 $f(0,0,0)$ 可行。若选第 i 个元素 v，则有 $f(i,j,k) \leftarrow f(i-1,j-1,(k-v) \bmod M)$，若不选，则有 $f(i,j,k) \leftarrow f(i-1,j,k)$。这个过程可以将第二维或第三维实现基于二进制位的并行，其时间复杂度在 w 位计算机上为 $\mathcal{O}((2M-1)M\frac{M}{w})$。

对于任何一个 $M = p_1 \cdot p_2 \cdots p_k$，其中 p_i 是可能相同的质数，如果依次让 $p = p_i$ $(1 \le i \le k)$ 执行上述过程 $2q-1$ 次，可以用交换论证的方法证明存在最优的分解顺序使得复杂度最小，此时 $p_1 \le p_2 \le \cdots \le p_k$。可以发现，质因数越多、质因数越小的 M 越容易找到解。

注意到随着 N 的增加 $[49.5\%N, 50.5\%N]$ 中的数字会越来越多，$[49.5\%N, \lceil N/2 \rceil]$ 中越可能存在容易计算的 M。设 $f(M)$ 表示从 $2M-1$ 个数中选出方案的最小代价，则可以选择区间中 $f(M)$ 最小的。最坏情况是 $N = 194$。

当 $[49.5\%N, 50.5\%N]$ 中存在整数时，必然包含 $\lceil N/2 \rceil$，因此判断无解只需要判断区间中是否存在整数。

D. 回文串

显然 $f(S[L..R]) \le R - L + 1$，所以答案不超过 $\frac{1}{12}N(N+1)^2(N+2)$，这是一个小于 2^{63} 的正整数。

但是 $f(S[L..R])$ 不总是等于 $R-L+1$，例如 $f(\text{abca}) = 3$。

注意到回文子串的数量不超过 N，我们不妨尝试计算 S 中每种回文串 T 的贡献。

很容易计算得出所有区间的长度之和是 $\binom{N+2}{3}$，不妨计算不产生贡献的区间 $[L,R]$，从全部长度中减去这些长度。

我们不妨将这些区间按照右端点 R 分类，设 T 在 S 中的出现位置从左到右依次是 $[l_1,r_1],[l_2,r_2],\cdots,[l_k,r_k]$，它们都满足 $r_i - l_i + 1 = |T|$。

对于 $1 \le R < r_1$ 的情况，我们有 $1 \le L \le R$，这些区间的贡献是 $\binom{r_1+1}{3}$。

对于 $r_i \le R < r_{i+1}$ 或 $r_k \le R \le N$ 的情况（不妨定义 $r_{k+1} = N+1$），我们有 $l_i < L \le R$，这些区间的贡献是 $\binom{r_{i+1}-r_i+|T|}{3} - \binom{|T|}{3}$，是关于 $|T|$ 和 $(r_{i+1}-r_i)$ 的多项式。

因此我们可以使用 Manacher 算法配合 Hash 或者直接使用 eertree 找出所有回文子串，使用后缀数组或者后缀自动机配合倍增算法识别出每个回文子串出现位置的右端点，通过启发式合并或者可持久化线段树合并或者其他数据结构维护 $(r_{i+1}-r_i)^e$ 的和，其中 $e = 1,2,3$。时间复杂度为 $\mathcal{O}(n \log n)$。

A. Product

Notice that array b has not been changed, so the useful elements in it will only be the minimum or the maximum.

Array a may be modified by erasing the minimum or maximum element, so the potentially useful elements include not only the minimum and the maximum but also the second minimum and the second maximum.

There are not so many test cases, so you can reduce the size of the a array to $\min(N, 4)$, reduce the size of the b array to $\min(M, 2)$, and then enumerate any possible situations.

Note that the answer may exceed 2^{32}, and Dr.Chen may choose to not use any magic.

B. Sort

Through the simulation of sample tests, it is easy to observe that the i-th element of the array after k times iterations is from the interval $[1, \min(i + k, n)]$ of the original array, so we can design an algorithm transforming the original array to the new array for any fixed parameter k in time complexity $\mathcal{O}(n \log n)$.

Here are two solutions that are accepted for this problem. One is to use a binary search based on the above algorithm because each iteration will help the array improve its preference in lexicographical order. However, it is easy to determine the lowest possible k because it is not difficult to match each element in these two arrays, so another solution is to find the minimum k and then check if it is possible. The time complexity of the former is $\mathcal{O}(n \log^2 n)$ while that of the latter is $\mathcal{O}(n \log n)$.

C. Generator

For a fixed M, it is obvious that there exists the minimum positive integer K_M such that there exists a subset of size M satisfying the sum of elements in the set is a multiple of M for any N integers ($N \geq K_M$) might include identical integers. For example, if there exists an integer appeared at least M times, a trivial solution can be constructed, which means $K \leq (M-1)^2 + 1$. In fact, there are kinds of proof for $K = 2M - 1$, which is called Erdös-Ginzburg-Ziv theorem. Here is one solution using counter-evidence with elementary number theory.

In case of $M = 1$, it is trivial.

In case of $M = p \cdot q$ ($p, q > 1$), one can from these $2M - 1$ integers select $2q - 1$ sets of size p satisfying the sum of elements in each set is a multiple of p, and then from these $2q - 1$ sets select q sets satisfying the sum of elements in the q sets is a multiple of M. Therefore, we only remain to solve the case that M is a prime.

In case that M is a prime, assume all the selections that pick from these $2M - 1$ integers M integers satisfy the sum of M elements is not a multiple of M (i.e. coprime to M), and then establish some contradiction. Let these $2M - 1$ integers as $a_1, a_2, \cdots, a_{2M-1}$, and $J = \{1, 2, \cdots, 2M - 1\}$, consider the formula:

$$S = \sum_{I \subset J, |I| = M} \left(\sum_{i \in I} a_i \right)^{M-1}$$

Based on assumption, we hold that $\gcd(\sum_{i \in I} a_i, M) = 1$. By Fermat's little theorem, it is known that $\left(\sum_{i \in I} a_i\right)^{M-1} \equiv 1 \pmod{M}$ and thus $S \equiv \sum_{I \subset J, |I| = M} 1 \equiv \binom{2M-1}{M} \pmod{M}$. In terms of Lucas' theorem, one can conclude $S \equiv \binom{1}{1}\binom{M-1}{0} \equiv 1 \pmod{M}$.

On the other hand, let's represent S as the sum of monomials of some a_i, and we know the coefficient of

$\prod_{k=1}^{t} a_{p_k}^{e_k}$ $(1 \le t \le M-1, \sum_{k=1}^{t} e_k = M-1)$ in the expansion is $\binom{2M-1-t}{M-1}\frac{(M-1)!}{\prod_{k=1}^{t} e_k!}$. By applying Lucas' theorem again, we have $\binom{2M-1-t}{M-1} \equiv \binom{1}{0}\binom{M-1-t}{M-1} \equiv 0 \pmod{M}$, so $S \equiv 0 \pmod{M}$, which is contradict with aforementioned conclusion. Hence, the assumption is false and there exists at least one selection satisfying the sum of its elements is a multiple of M.

Dynamic programming may help find a solution. Define $f(i,j,k)$ as the possibility of selecting j integers from the first i integers such that the sum of them is k in modulo M. The initial state is that $f(0,0,0)$ is true. We have $f(i,j,k) \leftarrow f(i-1,j-1,(k-v) \bmod M)$ if it selects the i-th element v, and $f(i,j,k) \leftarrow f(i-1,j,k)$ if not select. The second or the third dimension of this procedure can be parallelized in bit-level such that the time complexity on w-bit computer is $\mathcal{O}((2M-1)M\frac{M}{w})$.

For any $M = p_1 \cdot p_2 \cdots p_k$, where each p_i is prime that might be identical, if we process the procedure with $p = p_i$ $(1 \le i \le k)$ in order $2q-1$ times, we can prove by exchange argument that there exists an optimal order $p_1 \le p_2 \le \cdots \le p_k$ which approaches to sophisticated complexity. It shows that it is easy to find a solution in case that M has serveral small prime factors.

With the increase of N, the number of integers in $[49.5\%N, 50.5\%N]$ increases and the probability of the existence of some easy-to-solve integer M in $[49.5\%N, \lceil N/2 \rceil]$ also increases. Let $f(M)$ as the minmum cost to find a solution from $2M-1$ integers, and you can select M in the interval with the lowest cost to solve. The worst case is $N = 194$.

While there exists some integer in $[49.5\%N, 50.5\%N]$, $\lceil N/2 \rceil$ is already contained in that, so to check the existence of any solution is equivalent to check the existence of integer in that interval.

D. Palindrome

It is obvious that $f(S[L..R]) \le R - L + 1$ and thus the answer does not exceed $\frac{1}{12}N(N+1)^2(N+2)$, a positive integer that less than 2^{63}.

But $f(S[L..R])$ is not always equal to $R - L + 1$. For instance, $f(\text{abca}) = 3$.

Notice that the number of distinct palindromic substrings does not exceed N, so let's try to calculate the contribution of each palindrome T appeared in S.

The total length of all the intervals is easy to show as $\binom{N+2}{3}$. Let's eliminate the length of unused intervals $[L,R]$ from the contribution.

Let's order the intervals by their right endpoint R. Let all the appearance of T in S as $[l_1,r_1],[l_2,r_2],\cdots,[l_k,r_k]$ satisfying $r_i - l_i + 1 = |T|$.

In case of $1 \le R < r_1$, we know $1 \le L \le R$ and their contribution is $\binom{r_1+1}{3}$.

In case of $r_i \le R < r_{i+1}$ or $r_k \le R \le N$ (define $r_{k+1} = N+1$ without losing generality), we know $l_i < L \le R$ and their contribution is $\binom{r_{i+1}-r_i+|T|}{3} - \binom{|T|}{3}$, a polynomial of $|T|$ and $(r_{i+1} - r_i)$.

Therefore, we can apply the manacher algorithm with hashing or use eertree directly to detect all palindromic substrings, utilize suffix array or suffix automaton with a doubling algorithm to identify all the right endpoints of appearances of each palindromic substring, and maintain the sum of $(r_{i+1} - r_i)^e$ for $e = 1,2,3$ by heuristic merging sets, persistent segment tree with merging or other data structures. The time complexity is $\mathcal{O}(n \log n)$.