联考题解

T1 变换

算法一

从初始状态广搜，搜到所有数相等时停止。

期望得分 8 pts。

算法二

当 $i=j$ 时，操作相当于将 a_i 在二进制下左移一位。

考虑重复以下操作，将 a_1 在二进制下最低位 1 与 a_2 的最低位（唯一的）1 对齐：

- 当 $\operatorname{lowbit}(a_1)\neq\operatorname{lowbit}(a_2)$ 时，若 $\operatorname{lowbit}(a_1)<\operatorname{lowbit}(a_2)$，则令 a_1 左移一位，否则令 a_2 左移一位，直到 $\operatorname{lowbit}(a_1)=\operatorname{lowbit}(a_2)$。

对齐后若 a_1 是 2 的非负整数次幂，则已经有 $a_1=a_2$。否则令 $a_1=a_1+a_2$。

类似树状数组，考虑二进制位，可以证明操作次数为 $O(\log(a_1)+\log(a_2))$。

不需要写高精度。

结合 **算法一**，期望得分 20 pts。

算法三

首先一个简单的性质：

> *操作是线性的。*
> *那么对于一个操作序列，若其可以将 a_1,a_2 变为相同数字，那么该操作序列也可以将 ka_1,ka_2 变为相同数字，其中 k 为任意实数，反之亦然。*

有如下两种算法。

算法 3.1

由于上述性质，可以将 a_1,a_2 同时除以 $\gcd(a_1,a_2)$。这不影响构造。

可以证明，若 a_1,a_2 互质，操作结束时 a_1,a_2 一定都是 2 的非负整数次幂。

考虑通过一些乱搞做法将 a_1,a_2 中某个数变为 2 的整数次幂，然后执行 **算法**二。

随机数据下，该算法操作次数约为 200 次。需要写高精度。

算法3.2

抛弃部分分 2。

考虑有如下算法：

- 若 a_1, a_2 都是偶数，则令 a_1, a_2 同时除以 2。
- 若 a_1 和 a_2 只有一者为奇数，设 a_i 为奇数，则执行操作 `i i`，即令 $a_1 = 2a_1$。
- 若 a_1 和 a_2 都是奇数，不妨设 $a_1 > a_2$，则执行操作 $a_1 = a_1 + a_2$。

可以证明该算法执行次数不超过 $\log a_1 + \log a_2 + 2\log^2(\max(a_1, a_2))$。详见末尾 Proof1。

事实上可以构造数据使该算法操作次数达到 $O(\log^2 a)$。详见算法八。

不需要写高精度。不需要使用 `long long`。

结合 **算法二**，期望得分 32 pts。

可以证明，当 $\gcd(a_1, a_2) = 1$ 时，最后变成的数总为 2 的非负整数次幂。该性质在 **算法七** 中会应用到。

算法四

使用 **算法3.2** 将 a_1, a_2 变为相同数字。再考虑与 a_3 进行合并。

由于这里不能再将 a_1, a_2 同时除以 2。实现时需要写高精度。

随机数据下，该算法约执行 200 次操作。

结合 **算法三**，期望得分 40 pts。

算法五

考虑通过 **算法四**，从 2 到 n 枚举 i，依次将 $a_1, a_2, \cdots, a_{i-1}$ 与 a_i 进行合并。

由于 $a_i \le 1000$ 且数据随机，该算法执行操作远小于操作次数上界。

需要写高精度。

期望得分 48 pts。

算法六

注意到部分分 $n = 4096$，考虑分治。

考虑设计函数 `solve(l, r)` 实现将 $a_l, a_{l+1}, \cdots, a_r$ 变为相同数字。

递归解决 $[l, mid]$ 和 $[mid+1, r]$ 两区间，其中 $mid = \lfloor \dfrac{l+r}{2} \rfloor$。使用 **算法三** 将两部分合并。

实际测试中使用 **算法3.2** 更优。两算法都能通过该部分数据。

需要写高精度。

期望得分 $56 \sim 80$ pts。或许较好的实现可以用该算法得到满分。

算法七

考虑当 n 比较小时，使用 **算法六**。

当 n 比较大时，由于数据随机，期望存在 i 使 $\gcd(a_1, a_i) = 1$。

$O(n)$ 找到这样的 i，使用 **算法**三 将 a_1, a_i 中之一变为 2 的非负整数次幂。

用该数字将其它数字消成相同数字。具体地，设 i 满足 a_i 为 2 的非负整数次幂，重复以下算法:

- 枚举所有 a_k，若 $\mathrm{lowbit}(a_k) = \mathrm{lowbit}(a_i)$ 且 $k \neq i$，那么执行操作 $a_k = a_k + a_i$。
- 执行操作 $a_i = a_i + a_i$。

期望得分 $64 \sim 100$ pts。

算法八

考虑扩展 **算法3.2**。在 **算法六** 中使用该算法将该算法用于分治中合并两段，这里将该算法扩展为直接将所有数操作为相同数字。

分奇偶性。考虑重复以下算法直到 $a_1 = a_2 = \cdots = a_n$:

- 找到 $a_1, a_2, \cdots, a_n$ 中最小的奇数，设为 a_i。
- 枚举所有 a_j，若 a_j 是奇数且 $j \neq i$，执行操作 $a_j = a_i + a_j$。
- 执行操作 $a_i = a_i + a_i$。
- 此时必然所有数都是偶数，于是可以将所有数字都除以 2。

该操作得到的序列一定是可行的。

随机数据下该算法表现优秀。

笔者目前造出的最差数据如下:

- 设 n 为偶数，k 为非负整数，令 $\begin{cases} a_1 = a_2 = \cdots = a_{n/2} = 2^k + 1 \\ a_{n/2+1} = \cdots = a_{n-1} = a_n = 2^k + 2^{k+1} + 1 \end{cases}$。该算法操作次为 $O(nk^2)$ 级别。

该算法不需要写高精度。

期望得分 100 pts。

Proof1:

考虑当 a_1, a_2 都是奇数时，不妨设 $a_1 > a_2$，那么下一步操作为 $a_1 = a_1 + a_2$。

此时 a_1 为偶数，故下一步操作为 $a_2 = a_2 + a_2$。

此时 a_1, a_2 都是偶数，故将 a_1, a_2 同时除以 2。

即:

- 若 $a_1, a_2\ (a_1 > a_2)$ 都是奇数，则后两步操作可以看成 $a_1 = \lfloor \frac{a_1 + a_2}{2} \rfloor$。以下称该操作为操作一。
- 否则令偶数除以 2。以下称该操作为操作二。

操作过程中保证 a_1, a_2 都是正整数，于是操作一至多 $O(\log a_1 + \log a_2)$ 次。

每次执行操作一，$a_1 - a_2$ 恰缩小至原来一半，当 $a_1 - a_2 = 1$ 时总有一个数字为偶数，故不超过 $\log(a_1 - a_2)$ 步就会出现一个偶数，下一步会执行操作一。

于是总执行次数不超过 $\log a_1 + \log a_2 + 2\log^2(\max(a_1, a_2))$。

T2 密码

测试点 1-2

暴力枚举约数即可。

测试点 3-5

线性筛即可。原问题是 $1(x)=1$ 卷上 $\mathrm{id}^k(x)=x^k$，显然是积性的。时间复杂度 $O(A\times B)$。

测试点6-8

筛约数个数应该是一个经典的莫反问题。我们要解决的问题是：对 $\sigma_0(nm)$，如何将其拆解成不包含 n 和 m 的乘积形式。

不妨设 $f(n,p)=\sum_i[p^i|n\ \wedge\ p^{i+1}\nmid n]$，即质数 p 在 n 中的次数。考虑 n,m 的一个约数 d，考虑其单个质因子 p_i，其次数或被完全包含在 n 之中，即 $f(d,p)<f(n,p)$，或其次数完全包含 n。

考虑到上述特点，现在我们只需要枚举 i,j 分别为 n,m 的约数，并使 $\gcd(i,j)=1$，我们便可不重不漏的枚举 d。对于 $i\times j$ 的每个质因数 p，我们有如下关系：

$$f(d,p)=\begin{cases} f(i,p) & f(i,p)\neq 0\\ f(n,p)+f(j,p) & f(i,p)=0\end{cases}$$

容易看出，我们现在建立了 i,j 和 d 间的一一对应关系。那么我们直接反演即可。

$$
\begin{aligned}
&\sum_{i=1}^{A}\sum_{j=1}^{B}\sigma_0(ij)\\
=&\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{k|i}\sum_{d|j}[\gcd(k,d)=1]\\
=&\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{k|i}\sum_{d|j}\sum_{l|\gcd(k,d)}\mu(l)\\
=&\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{l|gcd(i,j)}\mu(l)\sum_{k|\frac{i}{l}}\sum_{d|\frac{j}{l}}1\\
=&\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{l|gcd(i,j)}\mu(l)\times\sigma_0(\frac{i}{l})\times\sigma_0(\frac{j}{l})\\
=&\sum_{l=1}^{\min(A,B)}\mu(l)\sum_{i=1}^{\lfloor\frac{A}{l}\rfloor}\sigma_0(i)\sum_{j=1}^{\lfloor\frac{B}{l}\rfloor}\sigma_0(j)
\end{aligned}
$$

线性筛 μ，剩余的部分可以数论分块。时间复杂度 $O(n)+O(T\sqrt{n})$。

测试点 9-10

用杜教筛处理各部分即可。注意到 $\sigma_0(x)=1(x)*1(x)$，那么构造杜教筛 $\sigma_0(x)*\mu(x)=1(x)$ 即可。注意用 `unordered_map` 或其他工具记忆化。

测试点 11-15

现在考虑如何来拆解 $\sigma_k(nm)$。因为该函数是积性的，我们直接对每个质因子讨论其答案即可。

考虑素因子 p，设 $f(n,p)=e_a$，$f(m,p)=e_b$，我们所求即为

$$\sum_{i=0}^{e_a-1}p^{ik}+p^{ke_a}\sum_{i=0}^{e_b}p^{ik}$$

利用和刚刚类似的思路，尝试把和式的两部分组合起来，并使 i,j 中某一项为 0。

$$
\begin{aligned}
&\sum_{i=0}^{e_a-1} p^{ik} + p^{ke_a}\sum_{i=0}^{e_b} p^{ik} \\
=&\sum_{i=1}^{e_a} p^{k(e_a-i)} + p^{ke_a}\sum_{i=0}^{e_b} p^{ik} \\
=&\sum_{i=1}^{e_a} p^{k(e_a-i)}\sum_{j=0}^{e_b} p^{jk}[j=0] + \sum_{i=0}^{e_a} p^{k(e_a-i)}[i=0]\sum_{j=0}^{e_b} p^{jk} \\
=&\sum_{i=0}^{e_a}\sum_{j=0}^{e_b}[\min(i,j)=0]p^{k(e_a-i)}p^{kj}
\end{aligned}
$$

还原，于是有

$$
\sigma_k(nm) = \sum_{x|n}\sum_{y|m}[\gcd(i,j)=1](\frac{ny}{x})^k
$$

其具有和前述内容类似的组合意义，不过是反着的。那么剩下的反演部分就和前面非常类似了，我们只给出最后的结果。

$$
\sum_{l=1}^{\min(A,B)} \mu(l)l^k \sum_{i=1}^{\lfloor \frac{A}{l} \rfloor}\sigma_k(i)\sum_{j=1}^{\lfloor \frac{B}{l} \rfloor}\sigma_k(j)
$$

现在可以线性筛处理了，注意到 $\mu \cdot id^k$ 是积性的（积性函数点乘完全积性函数仍是是积性的）。

测试点 16-25

这一部分的主要工作是构造 σ_k 和 $\mu \cdot id^k$ 的杜教筛。

对 σ_k，类似 σ_0，我们有 $\sigma_k * \mu = id^k$。而对 $\mu \cdot id^k$，我们有 $(\mu \cdot id^k) * id^k = e$，其中 $e(x) = [x=1]$。

也可以从狄利克雷生成函数的角度来推导。我们有

$$
\begin{aligned}
&\text{PGF } e(x) = 1 \\
&\text{PGF } 1(x) = \zeta(x) \\
&\text{PGF } \mu(x) = \frac{1}{\zeta(x)} \\
&\text{PGF } id^k(x) = \zeta(x-k)
\end{aligned}
$$

以及若 $\text{PGF } f(x) = F(x)$，那么 $\text{PGF } (f \cdot id^k)(x) = F(x-k)$。这样上面构造杜教筛的过程就显得相当自然了。自然数幂和的部分插值一下把多项式抓出来就可以了。那么最终的时间复杂度是 $O(n^{\frac{2}{3}}k + T\sqrt{n})$。

我们完整的解决了这个问题。

T3 序列

依然是蒯的，本题是`CF1423G Growing flowers`。有兴趣的同学可以去看看原题题解。

测试点 1-4

暴力即可。时间复杂度 $O(n^2)$。

测试点 5-6

因为并不是有启发意义的部分分，所以给的分值并不高，仅仅作为暴力的补充。用线段树维护区间覆盖操作，每次询问的时候重构序列，再直接 $O(n)$ 处理询问就好了。时间复杂度 $O(200n)$。

测试点7-12

比较重点的部分分。算上下面一些测试点一共给了 36 分。

考虑对于这类问题的经典转化：我们记录对每个位置而言，与其相同的数字上一次出现的位置，不妨对 i 位置我们记为 las_i。那么现在，我们所要求的便是

$$\sum_{i=1}^{n-k+1}\sum_{j=i}^{i+k-1}[las_j<i]$$

到这里可以写个巨大的分类讨论，用二维偏序做掉。但我们有更简单的办法。不妨把问题由计算单个 a_i 产生的贡献转化为一对 a_i 和 a_{las_i} 会使我们在每个长度为 k 的区间中需要减去的部分。

以数列 $\{1,2,3,2,5,6\}$ 为例。我们考虑在 2 和 4 位置上的 2 对于每一种询问会减去的部分。（本段下文的贡献均指需要减去的部分）。

- 对 $k=1,2$ 的询问，重复的数字不会产生任何影响。
- 对 $k=3$ 的询问，重复的数字会产生 1 的影响。
- 对 $k=4,5$ 的询问，重复的数字会产生 2 的影响。（对区间 $[1,4]$，$[2,5]$，$[1,5]$，$[2,6]$ 各产生 1 的贡献）。
- 对 $k=6$ 的询问，重复的数字会产生 1 的影响。

那么容易发现，对于一对 (i,las_i) 会产生贡献的区间，产生贡献的大小随着 k 增大一定会是先增大，然后维持不变，然后减小的。且增大和减小的部分一定是一段公差为 ±1 的等差数列。于是我们不妨用线段树来维护这些贡献。

记线段树上 $1\sim k$ 部分的和为 k 类型询问需要减去的大小。那么上述过程在线段树上也就是对区间 $[3,4]+1$，对区间 $[6,7]-1$。

这样我们就解决了不存在修改的情况，时间复杂度 $O(n\log n)$。

测试点 13-15

和 `5-6` 一样，把修改用线段树维护即可。

测试点 16-25

我们直接讨论正解的做法。`16-20` 的一部分是给想到正解但是不想写的人准备的（。

考虑维护连续一段相同的数字。现在我们考虑这样一个连续段对答案产生的贡献。

考虑在 $[3,5]$ 有连续一段相同的数字，而序列的总长度是 9。分别考虑 $[3,5]$ 中每一对相邻相同的数字，那么：

- 对 $[3,4]$，对 k 从 1 到 9 的贡献分别是 $\{0,1,2,3,3,3,3,2,1\}$。
- 对 $[4,5]$，贡献分别是 $\{0,1,2,3,4,4,3,2,1\}$。

将问题放到刚刚的线段树上，我们可以发现，同一段相同数字在线段树上的贡献可以被拆解为常数个等差数列。这样我们可以把相邻相同的一段数字放在一起处理，将这统称为一次操作。也就是说，每当我们执行一次区间推平的时候，我们只需要先处理掉被覆盖部分的答案，再用三次操作将新的贡献加入即可。这也是官方题解给出的做法。

不过，其实也有另外一种简单的处理方法。观察到每次覆盖时，只有 (i,las_i) 发生变化的位置的贡献才会发生变化。所以我们也可以直接处理被覆盖部分中每种连续相同段的两端的 las_i 变化，再处理新段的 las_i 变化即可。

那么，时间复杂度为什么是正确的呢？初始时刻我们有 $O(n)$ 个连续段，每次覆盖删除若干个连续段并加入三个连续段，而每个连续段仅会被删除一次，那么总连续段数目是 $O(n+m)$ 的。同理，las 的改变次数也是 $O(n+m)$ 的。

于是我们解决了这个问题，时间复杂度 $O(n+m)$。