

Solution

Wearry

Stay determined!

Bitcoin

注意到题目中的 K 比较小, 同时注意到题目名称提示了 BIT.....

考虑二进制第 K 位为 1 的数, 在模 2^{K+1} 意义下是连续的. 于是可以考虑对于每一个 K 维护这样的一个数据结构, 做加法时维护区间整体加标记即可.

时间复杂度 $O(KN \log N)$.

Exam

不妨设 $A \leq B \leq C, N = \max\{A, B, C\}$, 那么:

$$d(i \times j \times k) = \sum_{x|i} \sum_{y|j} \sum_{z|k} [(x, y) = 1][(y, z) = 1][(z, x) = 1]$$

$$\begin{aligned}
& \sum_{i=1}^A \sum_{j=1}^B \sum_{k=1}^C d(i \times j \times k) \\
&= \sum_{x=1}^A \sum_{y=1}^B \sum_{z=1}^C [(x, y) = 1][(y, z) = 1][(z, x) = 1] \lfloor \frac{A}{x} \rfloor \lfloor \frac{B}{y} \rfloor \lfloor \frac{C}{z} \rfloor \\
&= \sum_{x=1}^A \lfloor \frac{A}{x} \rfloor \sum_{d=1, (x,d)=1}^B \mu(d) \sum_{y=1, (x,y)=1}^{\lfloor \frac{B}{d} \rfloor} \lfloor \frac{B}{dy} \rfloor \sum_{z=1, (x,z)=1}^{\lfloor \frac{C}{d} \rfloor} \lfloor \frac{C}{dz} \rfloor
\end{aligned}$$

暴力计算上式可以做到 $O(N^2 \ln N)$ 的复杂度, 考虑优化, 不难发现:

- 对于 x 我们只关心它包含哪些质因子而不关心次数.
- 在不考虑互质的条件下这个式子是很好计算的.

定义:

$$f_x(n) = \sum_{i=1, (i,x)=1}^n \mu(i)$$

$$g_x(n) = \sum_{i=1, (i,x)=1}^n \lfloor \frac{n}{i} \rfloor$$

考虑已知 f_x, g_x 求 f_{xp}, g_{xp} (p 是质因子且 $(x, p) = 1$):

$$\begin{aligned}
f_{xp}(n) &= f_x(n) - \sum_{i=1, (i,x)=1}^n \mu(i)[p \mid i] \\
&= f_x(n) - \sum_{i=1, (i,xp)=1}^{\lfloor \frac{n}{p} \rfloor} \mu(i)\mu(p) \\
&= f_x(n) + f_{xp}(\lfloor \frac{n}{p} \rfloor) \\
g_{xp}(n) &= g_x(n) - \sum_{i=1, (i,x)=1}^n \lfloor \frac{n}{i} \rfloor [p \mid i] \\
&= g_x(n) - \sum_{i=1, (i,x)=1}^{\lfloor \frac{n}{p} \rfloor} \lfloor \frac{n}{pi} \rfloor \\
&= g_x(n) - g_x(\lfloor \frac{n}{p} \rfloor)
\end{aligned}$$

从 $x = 1$ 开始搜索遍历每一种可能的质因子集合, 记录每一层的 f, g 并使用它们推出下一层的 f, g 即可, 同时我们只需要维护其中 $O(\sqrt{N})$ 个位置的值, 复杂度 $O(N\sqrt{N})$.

Graph

考虑将联通块个数 T 的 k 次方看成多项式 $(x_0 + x_1 + \dots + x_T)^k$ 的展开. 那么展开的每一项相当于选择一些联通块然后任意排列:

$$T^k = \sum_{i=1}^k \binom{T}{i} \left\{ \begin{matrix} k \\ i \end{matrix} \right\} i!$$

考虑消去 T , 可以先枚举一个大小为 i 的联通块集合, 然后计算这个集合在多少张图中存在, 再乘上 $\left\{ \begin{matrix} k \\ i \end{matrix} \right\} i!$ 计算贡献即可.

因为 k 的范围比较小, 我们可以计算联通块个数为 $1 - k$ 的图的数量的生成函数, 同时不难计算答案的生成函数. 于是可以做到 $O(kn \log n)$ 预处理, $O(1)$ 询问.