zimpha's Contest 3

A. Almost Bobo Number

对于长度为 d 的数字（$d > 4$），$\underline{999\ldots999}898$（$(d-4)$ 个 9 后面有个 898）是合法的。

显然最后答案是某个前缀加上一段其他数字，并且在所有合法前缀中只有最长的前缀是有用的。考虑 s 是某个前缀加上某个字符，t 是合并重复数字后的字符串，我们找到 t 的一个长度小于等于 $\frac{|t|}{2}$ 的最长 border，设为 l，如果 $|t| - 2l \le d - |s|$，那么这个 s 是合法的。找到最长的之后，从这个最长开始，重复上面的过程应该就可找出最长的串了。

问题变成如何快速找到最长的那个 border，考虑到其实 border 的长度构成了 $O(\log n)$ 个等差数列，于是用 KMP 就可以 $O(1)$ 求出答案。

B. Connected Spanning Subgraph

定理：连通图的连通生成子图的数量是奇数当且仅当是二分图。

证明：考虑计算 $\sum_{S\subseteq E} 2^{c(S)} \bmod 4$，其中 $c(S)$ 是 S 的生成子图的连通块数量。因为 $x > 1$ 时 $2^x \equiv 0 \pmod 4$，所以不连通的子图贡献 0。连通的生成子图贡献 2，所以答案是上式除以 2.

上式的直观解释是，选择边的子集 S，对 V 中的连通块黑白染色。交换求和号后等价于，把点染成黑、白两种颜色，要求边的两端同色。

对于一种顶点的染色方案，对应的边的方案数是 2 的两端同色的边的数量次方。如果加上染色方案 C 和 C 的反，那么只要有同色的边，贡献就是 0.

因此，只需要统计所有边都不同色的顶点染色方案数。对于连通二分图，恰好只有 2 种。对于非二分图，不存在。

C. Power of Power Partition Function

可以简单的算出

$$b_m(n) = \begin{cases} b_m(n-1) + b_m(\frac{n}{m}) & n \bmod m = 0 \\ b_m(n-1) & n \bmod m \neq 0 \end{cases}.$$

然后有可以发现 c_m^k 的 k 阶差分之后是:$b_m(0), \underbrace{0, \ldots, 0}_{m-1 \text{ zeros}}, b_m(1), \underbrace{0, \ldots, 0}_{m-1 \text{ zeros}}, b_m(2), \underbrace{0, \ldots, 0}_{m-1 \text{ zeros}}, \ldots.$

于是直接套用Abel 求和变换就可以了。每次可以把 n 缩小成 $\frac{n}{k}$。

对于给出的两个数列 a_n 和 b_n，如果想要求 $S_N = \sum\limits_{n=0}^{N} a_n b_n$，令 $B_n = \sum_{k=0}^{n} b_k$，那么：

$$\begin{aligned}
S_N &= a_0b_0 + \textstyle\sum_{n=1}^{N} a_n(B_n - B_{n-1}), \\
&= a_0b_0 - a_0B_0 + a_NB_N + \textstyle\sum_{n=0}^{N-1} B_n(a_n - a_{n+1}), \\
&= a_NB_N - \sum_{n=0}^{N-1} B_n(a_{n+1} - a_n)
\end{aligned}$$

D. Line Counting

下面方法应该对很多凸集（包括高维）都是适用的。

令 $C_q(n)$ 表示恰好经过 q 个格点的格点线段数目，令 $l_{\geq q}(n)$ 表示经过至少 q 个点的格点直线的数目，那么 $l_{\geq q}(n) = C_q(n) - C_{q+1}(n)$，证明如下：

令 L 是答案集合，考虑 $l \in L$，且这条直线恰好经过了 p $(p \geq q)$ 个格点，那么这条直线包含了 $p - q + 1$ 条 q-格点线段。同时，这条直线包含了 $p - q$ 条 $(q+1)$-格点线段。令 $N(r, l)$ 表示 l 中的 r-格点线段的数目，那么 $N(q, l) - N(q+1, l) = 1$。于是：

$$\begin{aligned}
l_{p \geq q}(n) &= \sum_{l \in L} 1 \\
&= \sum_{l \in L} (N(q, l) - N(q+1, l)) \\
&= \sum_{l \in L} N(q, l) - \sum_{l \in L} N(q+1, l) \\
&= C_q(n) - C_{q+1}(n)
\end{aligned}$$

只要求出 $C_q(n)$ 就好了，这个十分简单。考虑枚举一个端点 (x, y) 以及向量 (i, j)，显然要求 $\gcd(i, j) = q - 1$，而且 $x + y + i + j \leq n + 1$。这个只要枚举 $d = i + j$，就可以随便推出公式了。

最后可以得到 $l_{\geq q}(n) = 3 \sum\limits_{i=1}^{\lfloor \frac{n-1}{q-1} \rfloor} \phi(i)(g(n-(q-1)i) - g(n-qi))$，其中 $g(n) = \frac{n(n+1)}{2}$，如果 $n \geq 0$，否则 $g(n) = 0$。把这个式子展开，可以发现只需要求 $\phi(i), i \cdot phi(i), i^2 \cdot \phi(i)$ 的前缀和就好了。

E. Maximum Flow

可以发现每一个极小割都是割一个上边界，一个下边界以及中间连续的一段，所以只需要枚举某个边界维护另一个边界可能的位置的最小值，另外直接跑最短路也是能解决的。

F. Rectangles Inside Rectangle

考虑一维问题（给出一堆区间 $[l_i, r_i]$），很容易写出一个 dp 方程，令 $dp(r_i)$ 是算到 r_i 的答案，考虑第 i 区间必选，那么 $dp(r_i) = dp(l_i) + w_i$，如果不选，那么 $dp(r_i) = dp(prev(r_i))$，其中 $prev(x)$ 是 x 前面最近的一个区间端点。

上面做法可以很轻易地推广到二维。以对于一个矩形 i，令 i_t 和 i_b 是矩形的上下边界。那么我们可以得到 $O(n^2)$ 个边界，总共可以分成三类：

1. 矩形的某个上下边界，也就是 i_t 或者 i_b；
2. 考虑两个不在同一侧且不想交的矩形 i 和 j，如果 $j_b \le i_t \le j_b$，那么延长 i_t 直到碰到矩形 j 为止，然后沿着边界往下走直到碰到 j_b，之后沿着 j_b 到另一侧。
3. 考虑两个不在同一侧且不想交的矩形 i 和 j，如果 $j_b \le i_b \le j_b$，那么延长 i_b 直到碰到矩形 j 为止，然后沿着边界往下走直到碰到 j_b，之后沿着 j_b 到另一侧。

然后，对于每个边界 δ，我们也可以唯一算出一个 $prev(\delta)$，表示 δ 和 $prev(\delta)$ 之间没有其他的边界。显然，这个可以随便 $O(n^2)$ 算出来。之后转移就方便了：1. 对于 i_t，如果不选 i，肯定从 $prev(i_t)$ 转移过来；如果选 i，需要找到一个不在同一侧且不相交矩形的 j，使得 $i_b \le j_t \le i_t$，然后从 (i_b, j_t) 转移过来。找不到的话，直接从 i_b 转移过来就好了。2. 对于 i_b，直接从 $prev(i_b)$ 转移过来即可。3. 对于 (i_t, j_b)，和 1 类似，同样分选或者不选转移一下就好了。4. 对于 (i_b, j_b)，和 2 类似，直接从 $prev$ 转移。

G. Cute Panda

显然可以直接建出一个图，然后跑一下网络流。但是这样应该会 TLE，考虑最大流 = 最小割，然后就可以得到一个线性的 dp。

H. Order-Preserving Partition

如果知道一段数的数量和最大值和最小值，就可以判断这个段是否合法。

对于 $Q = 1234$ 的 case，可以从左往右对段数 dp 一下得到答案。

对于 $Q = 21xx$ 和 $Q = 41xx$ 的 case，枚举 1 所属的段的最大值，那么第一段可以通过维护前缀信息来判断合法性，第三段和第四段拼接起来是合法的, 于是维护一下合法的第四段的数量即可，这个可以使用 two pointers 的技巧维护。

对于 $Q = 1324$ 和 $Q = 3124$ 和 $Q = 3142$ 的 case, 考虑数前两段的划分方案，枚举前两段的长度和，由于两段的分隔位置是唯一的 (因为相邻两段都不连续, 只有整个前缀是合法的情况才会更新分隔位置), 维护一下第二段的参数，那么这个就可以数了. 同理数出后缀两段的划分方案，和前缀合并一下即可。

其他的 case 都可以通过把序列翻转或取反转化到以上的 case, 时间复杂度 O(n).

I. Prime Tree

可以观察到：如果 $T = A \cdot B$ 并且 $A, B \ne \mathbf{1}$，令 n_Y 是 B 的根的每个儿子对应的子树的最大节点数。考虑 T 的根的每个儿子对应的子树，那些节点数小于等于 n_Y 的子树和 B 的根的每个儿子对应的子树是一样的。可以反证法证明这个东西。

然后，用上面的观察可以证明这个分解是唯一的。也就是说如果 $T = T_1 \cdot T_2 \cdots T_k$，令 $S_i = T_i \cdot T_{i+1} \cdots T_k$，只要找出每个 S_i 就好了。

首先找出每个子树的 hash 值，利用上面的观察，一个简单的暴力就是枚举 n_Y，然后找出 B，之后线性 for 一下每个节点判断这个 B 是不是这个节点的一个 factor 就好了（同样只要检查直接儿子即可）。这样复杂度是 $O(\sum\limits_{d|n} \frac{n}{d})$，和线性差不多。

线性的做法：先用线性 hash 求出每个节点的 hash 值，可以发现，S_i 一定是 T 的某个子树，那么可以先把 T 的所有子树按照大小和 hash 值排序，把每个子树根下面的儿子按照子树大小排序，然后对于每个点 u 构造一个序列 s_u，存 u 的儿子的 hash 值。你考虑从大到小枚举子树 u，u 是一个 S_i 当且仅当，那些比 u 大的子树的 s_v 的最长公共前缀恰好是 s_u。于是用些字符串匹配的技巧就可以做到线性了。

J. Hamiltonian k-vertex-connected Graph

对于 $k = 1$，显然是无解的。否则，先构造一个长度是 n 的环，这样就得到了一个点连通度为 2 的图。

如果 k 是奇数，对于每个 i $(1 \le i \le \lfloor \frac{n+1}{2} \rfloor)$，加入一条边 $(i, i + \lfloor \frac{n}{2} \rfloor)$，这样可以得到一个点连通图度为 3 的图。

接下来，对于每个 i $(1 \le i \le n)$，加入边 $(i, i + j)$，其中 $2 \le j \le \lfloor \frac{k}{2} \rfloor$。如果 $i + j > n$，需要自己调整下下标。