HNCPC2024题解

A. 贪吃蛇 (Maxzz)

~~本意是一道easy-medium题~~

枚举任意两点形成的有向边，选择夹角不超过90度的其他有向边得到一个有向无环图，图上的最长路即为答案，复杂度 $O(n^4)$

B.HolyK's Land (HolyK)

轻重链剖分后沿重链按先重儿子后轻儿子顺序标号，每个选项代表的点集可以拆成若干端重链的连续一段标号加上LCA和LCA的父亲。离线后，扫描线对标号序列染色，树状数组维护每个颜色的点的数量。

C. easy math (binarycopycode)

可以对乘积式子取log，然后比大小。

可以使用c++写高精度，或者使用python和java直接大数计算通过。

也可以把2024当成2048，通过2的幂次比较，因为 $(2024/2048)^{50} > 0.5$。

D. Too much Noise (alocytus)

推式子，把最大公约数的条件转换一下，得到求和是
$2\sum_{i=1}^{n}\sum_{j<i}\sum_{T|b_i,b_j}|a_ib_ib_j/T^2-a_jb_ib_j/T^2|f(T), f(T)=\sum_{k|T}\mu(k)k^2.$

f 可以预处理，考虑如何处理前面的求和。

首先按照 a_i 排序，这样可以把绝对值去掉。先枚举 i，再枚举 b_i 的所有因子 d。记录两个前缀和 $S^{(i)}_{1,d}=\sum_{j<i,d|b_j}b_j/d, S^{(i)}_{2,d}=\sum_{j<i,d|b_j}a_jb_j/d$，那么上述求和就是

$2\sum_{i=1}^{n}\sum_{d|b_i}(a_iS^{(i)}_{1,d}-S^{(i)}_{2,d})(b_i/d)f(d)$. 每次更新 $S^{(i+1)}_{1,d}=S^{(i)}_{1,d}+b_i/d, S^{(i+1)}_{2,d}=S^{(i)}_{2,d}+a_ib_i/d.$

预处理线性筛 $O(M)$，枚举所有因子用埃氏筛 $O(M\ln M)$. 然后这个范围内因子总数最大为 160. 求和的总共循环次数上界为 $160N$.

题目的背景来自本届 CCBC15 的大海捞针.

E. 拼接串 (binarycopycode)

注意到 $1\le a_i\le 18$，设全部数字都恰好出现一次的状态为 $m=2^{18}-1$。

所以可以维护 $f[0\le s\le m]$ 表示出现数字的集合为 s 时，所能得到的最大长度。从每个位置开始向右for循环，最多向右找18个数字没有重复的。

然后可以通过 f 求得 $dp[0 \le s \le m]$ 表示出现数字的集合是s的子集时，所能得到的最大长度。从小到大枚举集合，然后再枚举 s 每一位进行DP转移即可。

最后答案就是枚举所有的 集合状态 s ，答案就是 $\max\{dp[s] + dp[m\ xor\ s]\}$ 。

F. 阅读理解 (TaoQian)

代码求的是一个无向连通图的广义圆方树。由于节点编号无关紧要，一棵广义圆方树满足：叶子节点均为圆点；圆点和方点交替。只要满足上述条件的树就在本题中被称为“合法的”。同时，原图的割点个数就是度大于 1 的圆点个数。

考虑树形 DP，记 $f[u, i, 0/1/2]$ 表示 u 为方点，子树 u 内已经确定了 i 个割点，u 有 $0/1/\ge 2$ 个圆点儿子与之相连且不删去的方案数；记 $g[u, i, 0/1/2]$ 表示 u 为圆点，子树 u 内已经确定了 i 个割点，且 u 有 $0/1/\ge 2$ 个方点儿子与之相连且不删去的方案数。

由于一个圆点 u 目前有 0 个方儿子时，父亲与之相连是不会增加割点个数；圆点 u 目前有 ≥ 2 个方儿子时，父亲与之相连也不会增加割点个数（因为 u 已经是割点）；但 u 目前有 1 个方儿子的时候，父亲与之相连会增加割点个数。

同时，对于一个方点，目前有 $0, 1$ 个圆儿子时，父亲与之之间的边不能删去，否则得到的树的叶子为方点；目前有 ≥ 2 个圆儿子时可以删去父亲与之之间的边独立成一棵树。

由此可以进行树形背包。显然割点的上界为 n，因此时间复杂度 $\mathcal{O}(n^2 + q)$。

具体转移如下：

$$
\begin{aligned}
A_0 &= f[v, j, 2] + g[v, j, 0/1/2] \\
A_1 &= g[v, j, 0/2] + g[v, j-1, 1] \\
f'[u, i+j, 0] &= f[u, i, 0] \times A_0 \\
f'[u, i+j, 1] &= f[u, i, 1] \times A_0 + f[u, i, 0] \times A_1 \\
f'[u, i+j, 2] &= f[u, i, 2] \times (A_0 + A_1) + f[u, i, 1] \times A_1
\end{aligned}
$$

$$
\begin{aligned}
B_0 &= A_0 \\
B_1 &= f[v, j, 1] + f[v, j, 2] \\
g'[u, i+j, 0] &= g[u, i, 0] \times B_0 \\
g'[u, i+j, 1] &= g[u, i, 0] \times B_1 + g[u, i, 1] \times B_0 \\
g'[u, i+j, 2] &= g[u, i, 2] \times (B_0 + B_1) + g[u, i-1, 1] \times B_1
\end{aligned}
$$

选择原树的任意一个非叶子节点为根，则对于叶子节点，初始 $g[u, 0, 0] = 1$，其余均为 0。

答案为 $f[rt, x, 2] + g[rt, x, 0/1/2]$。

可能需要特判一下 $n = 1$ 和 $n = 2$ 之类的边界情况。

注意到割点个数小于最大独立集。先求出最大独立集大小即可，可以省一半的空间。与此同时，可以利用重链剖分的思想，优化效果玄学。

如果时间上被卡常，不妨试试 short。

G. Utakotoba (HolyK)

注意到操作是可逆的，AB两个数组的线性基相同，因此问题转换成将数组通过若干操作变成线性基，按位考虑即可

H. 经文 (TaoQian)

考虑dp。令 $f_{i,j,p}$ 表示前 i 个字符，已经完成 j 个 s 的匹配，且第 $j+1$ 个匹配进度为 p 的情况数。

每次枚举下一位填的字母 c。匹配成功则直接递推，匹配失败则暴力枚举找出失配状态即可。

可以看出，我们要找的就是对于原串的任意一个前缀s'，其最长公共前后缀的长度。这正是KMP算法的核心——求解kmp数组（或者叫做next数组）。

求出kmp数组后，对于 s 中的每一位 i，枚举它后一位如果填字母 c 并且失配，会转移到哪里即 $nxt_{i,c}$ 。其本质就是kmp模式串匹配。如果 $c=s[kmp_i+1]$，那么可以转移至 kmp_i，否则令 $i=kmp_i$ 递归寻找，直到$kmp_i=0$。

这样预处理的复杂度为 $O(26|s|^2)$，可以通过本题，但事实上可以降至 $O(26|s|)$，只需做一步小处理即可。

我们发现如果一个字符反复失配（即递归层数很大），其中有很多重复搜索。实现上可以反过来递推，即先用 $nxt(kmp_i)$ 更新 $nxt(i)$，再对 kmp_i+1 处的成功匹配情况进行修正即可。注意到如果有两个或更多成功匹配的情况，先选较长的一处一定是更优的，因为如果再次失配还能转移回较短的一处。

这样做到了预处理复杂度 $O(26|s|)$，总复杂度 $O(26nk|s|)$.

I. 数据检索系统 (binarycopycode)

按照题面模拟即可

J. Beautiful Sequence (Maxzz)

如果值域为 $[l,r]$ 的 beautiful seq 在两个排列上同时出现，则值域内其他短的 beautiful seq 也满足条件。线段树维护值域为 $[l,r]$ 子序列 hash 值，two pointer 从小到大枚举 r 得到每个 r 对应的最小 l，$sum(r-l)$ 即可

K. 渡劫 (TangRain)

先不考虑法宝，假设初始传送在 i 号岛屿，最终在 j 号岛屿上渡劫成功，那么所需要的能量是 $dis(i,j)+a_j$。路径是双向的，所以也等价于 $a_j+dis(j,i)$ 。

由于所有点都可以为初始点，为了方便处理，考虑额外建一个超级源点 s 与所有岛屿相连，其中 s 到每个岛屿的边权为 a_j ，前面的式子可以化简成 $dis(s,j)+dis(j,i)=dis(s,i)$ 。

至此，这道题实际上在求 s 到所有点最短路的最大值。（部分写法需要注意考虑传送在 i 并在 i 原地渡劫的情况。）

对于额外一次的法宝，使用分层图进行建图（层数为2）。在边 $u->v$ 使用法宝，就在第一层的 u 到第二层的 v 连一条边权为 0 的边。超级源点连接第一层图。

最后跑一遍最短路从第二层图中取答案即可。