A.减肥计划

- 显然队首的权值单调不减，因此被淘汰放入队尾的人不可能再后续的游戏中获胜。
- 当游戏进行n轮时，队列中权值最大的人必然会到队首，此时队首不可能输，因而只需要考虑最大值出现以前的那些人有没有可能获胜即可。
- 对于第$i(i>1)$个人权值为a_i，由于其到达队首时已经打败了1个人，其想要连续获胜k轮的条件是其后面有连续$k-1$个人的权值不大于a_i，即$\max_{j\leqslant i+k-1}a_j\leq a_i$。
- 对于第1个人，其一开始没有打败任何人，因此其想要连续获胜k轮的条件是其后面有连续k个人的权值不大于a_i，即$\max_{j\leqslant k+1}a_j\leq a_i$。
- 因为每轮游戏必然淘汰一个人，而游戏最多进行n轮，因此直接模拟或者使用单调栈求出距离每个人最近且权值大于它的位置后从左到右扫一遍进行判断即可。
- 需要注意当$k\geq n$时，除了权值最大的人，其他人都不可能成为最终赢家，因此直接输出第一个最大值的位置即可。
- 时间复杂度：$O(n)$，空间复杂度：$O(n)$。

B.修建道路 II

- 选定任意一个点作为树的根，点权变成边权，并且用一个 按 dfs 排序的set 来维护清单的城堡编号。
- 增加节点时，寻找节点按 dfs 序排序的前继与后继节点，总答案减去两个节点的路径权值，并且分别加上这个节点到前继、后继节点的路径权值。
- 删除节点时，进行相反的操作就好了。
- 每次输出答案时，还要加上清单中所有节点 lca 的节点权值。
- 时间复杂度大致为：$\Theta(nlog^2n)$

C.测量学

- 签到题。
- 对于所有输入 r，从外围到它的距离和它到外围的距离都是 $(R-r)$，绕轨道走的距离为 $r\cdot\theta$，故答案为 $\min\{2(R-r)+r\theta\}$。
- 注意点1：θ 的范围是$[0,2\pi)$，由于可以顺时针或逆时针走到，所以需要将 θ 和 $2\pi-\theta$ 取 min
- 注意点2：R 也对答案有贡献，故 $2(R-R)+R\theta=R\theta$ 也对答案有贡献。
- 注意点3：只取最小或最大的半径计算答案，都是错误的。

D.Devil May Cry

算法1

- 考虑状压DP，设 $f(i,S)$ 表示用前 i 种技能填入技能序列的位置集合为 S 时的方案数。
- 设 T_i 表示第 i 个技能不能放的位置的集合。
- 枚举第 i 个技能放置的位置集合，可得转移 $f(i,S)=\sum\limits_{T\subseteq S}[|T\cap T_i|==0][|T|\le lim_i]f(i-1,S-T)$
- 时间复杂度 $O(m3^n)$

算法2

- 考虑使用集合幂级数优化，设 $F_i=\sum\limits_{S\subseteq U}[|T_i\cap S|==0][|S|\le lim_i]x^Sy^{|S|}$
- 其中 U 表示位置的全集。
- 求出 $F=\prod\limits_{i=1}^{m}F_i$ ，则 $[x^Uy^n]F$ 就是答案。
- 此处的乘法表示子集卷积，单次复杂度 $O(n^22^n)$ ，连乘总复杂度 $O(mn^22^n)$

算法3

- 考虑使用集合幂级数的 $\ln$ 和 $\exp$ 变换来求连乘，$F=\exp(\sum\limits_{i=1}^{m}\ln(F_i))$
- 需要先进行 m 次集合幂级数的 $\ln$ ，时间复杂度 $O(mn^22^n)$
- 然后是集合幂级数连加，时间复杂度 $O(mn2^n)$
- 最后进行一次集合幂级数 $\exp$ ，时间复杂度 $O(n^22^n)$
- 总时间复杂度仍为 $O(mn^22^n)$

算法4

- 注意到算法三中，第一步求所有集合幂级数的 $\ln$ 变换为复杂度瓶颈。
- 思考集合幂级数 $\ln$ 的求法，首先做一次 **FWT** 变换，然后对每个集合的占位多项式求 $\ln$ ，最后再逆变换。
- 由于这里最后还要 $\exp$ 回来，$\exp$ 的步骤也类似，所以可以把 $\ln$ 之后的逆变换和 $\exp$ 前的正变换抵消掉。
- 令 $\hat{F_i}$ 为 F_i 正变换之后的结果。
- 只需要求 $\ln([x^S]\hat{F_i})$

$$\begin{aligned}\ln([x^S]\hat{F_i})&=\ln\sum_{T\subseteq S}[x^T]F_i\\&=\ln\sum_{T\subseteq S}[|T\cap T_i|==0][|T|\le lim_i]y^{|T|}\\&=\ln\sum_{t=0}^{lim_i}y^t\sum_{T\subseteq S-T_i}[|T|==t]\\&=\ln\sum_{t=0}^{lim_i}\binom{|S-T_i|}{t}y^t\end{aligned}$$

- 可见 $\ln([x^S]\hat{F_i})$ 只与 lim_i 和 $|S-T_i|$ 的值有关系，所以第一步求 $\ln$ 只需要预处理 $O(n^2)$ 个不同的占位多项式就行了，复杂度 $O(n^4)$ 。
- 第二步和第三步复杂度不变，故总时间复杂度为 $O((n+m)n2^n)$ ，优化至此，即可通过本题。

E.睡觉

- 也是偏签到的一道题目，难点可能在于分类讨论和一些细节。
- 首先需要把数组备长，模拟一轮，如果过程中出现连续 t 秒都小于等于 k 了，就符合条件。不倍长的话显然会出现错误。
- 然后需要一些分类讨论，当模拟一轮后，清醒值降低了，那显然是 YES ，因为经过无限长的循环后，清醒度总会降到 k
- 如果模拟一轮后清醒度升高了，并且过程中没有出现连续 t 秒都小于等于 k ，那显然是 NO
- 如果模拟一轮后清醒度不变，此时还有一种额外的情况，t 很大，并且整个过程中清醒度都小于等于 k ，这种情况显然也是 YES

F.位运算谜题

- 首先不难证明一个性质：$A \oplus B = (A|B) \oplus (A\&B)$，然后，可以依此性质将输入分成三组，每组都是一对数（比如 a 和 b）的三种运算的结果（$a \oplus b, a|b, a\&b$）。
- 接下来要考虑如何去分组，因为保证存在解，所以可以随机化，首先第一组第一个数是任意的，可以直接取输入的第一个数，第一组第二个数随机一个下标得到，第三个数直接根据上面的性质算出来，第二个数有至少 $\frac{2}{8}$ 的概率跟第一个数在同一组，也就是说我们有至少 $\frac{1}{4}$ 的概率得到正确的第一组。第二组可以用类似的方法，但是正确概率提升到了至少$\frac{2}{5}$，而第三组，只要前两组确定，第三组必然确定。
- 这样我们每次随机有至少 $\frac{1}{10}$ 的概率得到正确的分组，也就是期望随机次数不超过 10。
- 接下来考虑，得到分组之后，如何去还原三个数，首先我们要确定每组的三个数，哪个是异或的结果，哪个是与的结果，哪个是或的结果。显然，最大值一定是或的结果，但与和异或的结果不好确定，因此我们还要花 $2^3 = 8$ 次枚举去确定哪个是异或，哪个是与。注意到这步枚举可以不显式写出，只需要结合到上一步一起随机即可，这样不会增加代码量。
- 枚举好之后，可以把三个数分别看成是三个集合，异或是对称差运算，或是并运算，与是交运算，再画个韦恩图可以轻松推导出 a, b, c。最后把 a, b, c 代入验证，如果结果正确则输出，否则重新随机，这样我们的期望随机次数不超过 80。
- 本题亦可以搜索，但可能会有较为麻烦，可能被卡等缺点。

G.排队打卡

- 签到题，把事件按时间排序，接着按事件模拟即可。
- 模拟过程中计算出每个事件时刻，队列中的人数，校验在 YahAHa 醒来的时候人数是否相符，并计算每个事件时刻所需的排队时间即可。

H.提瓦特之旅

- 考虑对于每次询问，可以在 bfs 过程中带着 w_i 的信息查询最短路，这样的复杂度是 $O(q \times n \times n)$ 会导致超时。
- 如果能在询问前提前处理信息 $f[i][j]$ 表示从 1 走到 i 并且过程中经过了 j 个点的最短路，在查询的时候答案 $g[t][j]$ 即为 $f[t][j] + \sum_{i=1}^{j} w[j]$，这样每次询问的复杂度为 $O(n)$，预处理复杂度为 $O(n \times m)$，可以轻松通过此题。

I.宠物对战

- 比较签到的一道题，dp的形式比较明显，$f[i][0/1]$ 表示组成询问串的前 i 位，结尾的串属于 A/B 类串的最短步数。
- O(n)的时间枚举转移，每次枚举一个比 i 小的 j ，当 $j+1$ 到 i 的子串在 A/B 类串中，$f[i][0/1]=min(f[j][1/0]+1)$ 。
- 可以对 A/B 类串分别建 Trie 树或者采用字符串 hash 来维护上面的条件
- 时间复杂度 O($\sum|a_i|+\sum|b_i|+|S|^2$)

J.瑞士轮

- 首先由于期望的线性性，整份预测表中预测的正确数期望即为每个位置预测的正确数期望之和。
- 因此只需要算出每个队伍在 0 - 2，1 - 2，2 - 1，2 - 0 每种比分下的概率即可。
- 乍一看根据这个赛制，三十二支队伍总共需要有四十场比赛，2^{40} 的枚举显然是行不通的。
- 而由于后面的比赛对阵状况又取决于前面的比赛结果，因此也无法通过折半搜索优化。
- 但是仔细观察或者模拟一下会发现，我们把队伍四个四个分为一组，一个队伍的所有比赛对阵，只会在这四个一组中产生。
- 例如 1、2、3、4 号，打完第一轮会产生两个 1 - 0 和两个 0 - 1，这两个 1 - 0 和两个 0 - 1 按编号排序后显然是相邻的，之后打完第二轮产生一个 2 - 0 直接晋级，产生一个 0 -2 直接淘汰，两个 1- 1 又产生一次对阵。
- 因此在四个队伍一个的分组中，总共产生五次比赛，可以考虑分类讨论算出结果，也可以直接 2^5 枚举比赛结果然后模拟计算概率。

K.区间和

- 由于所有数字的总和不超过 10^6 ，考虑维护出区间和为每一种数字的区间个数。
- 考虑对 a_i 数组做前缀和，得到 $sum_i(0 \leq i \leq n)$ ，对 sum_i 求 cnt 数组，$cnt[sum_i]$ 表示大小为 sum_i 的前缀和的出现次数。 接着对 cnt 自己对自己做减法卷积，卷积的结果 $f_i(i > 0)$ 就表示区间和为 i 的区间个数。以下将作出说明。
- 对于区间 $[l, r]$ ，区间和即为 $sum[r] - sum[l-1]$ ，不妨设这个值为 x ,区间和为 x 的区间个数 num_x 即为满足 $sum[r] - sum[l-1]$，$l \leq r$ 的 (l, r) 对数。第一部分的限制是一个减法卷积的形式，而由于 $a_i \geq 0$ ，所以保证了当 $sum[r] - sum[l-1] > 0$ 时，r 也会大于 $l-1$ ，即 $r \geq l$ 。因此直接对 cnt 本身做减法卷积就能得到区间和大于 0 的所有结果。
- 而对于区间和等于 0 的部分，由于存在 $sum[i] - sum[i]$ 这种错误的转移，可以在卷积的结果中，扣除掉这一部分的转移，也可以单独算出区间和为 0 的区间个数。
- 由于区间和为 x 的区间个数可能会很大，超过 int 的范围，因此卷积需要采用 FFT 或者大模数的 NTT 等算法，即可通过此题。

L.彩色的树

- 极限情况下，需要求出所有节点的答案。
- 考虑 dfs 的过程中，维护以该节点为根的子树内的答案。用一个 cnt 数组记下每种颜色出现的次数，并通过 cnt 数组维护不同颜色的个数。
- 在考虑以 x 为根的子树时，假设已经得到 x 的儿子为根的子树信息，只需将儿子的信息上传，扣掉与 x 距离为 $k+1$ 的点的贡献。由于 k 是固定的，可以采用树上倍增求出子树内与自己距离为 k 的所有节点，也可以在dfs的过程中，记录所有祖先，直接维护。
- 此时只需要考虑不同儿子的信息如何合并，这里可以用多种方法维护：dsu on tree、线段树合并或者是对每个节点开一个map（multiset），然后采用普通的由小往大合并的启发式合并