01

- 循环位移这类东西倍长会更加好写。
- 哈希做法：可以先把 A 的所有本质不同循环位移的哈希值丢进 $\mathrm{map}/\mathrm{set}$ 里，然后再依次检查 B 中长度为 $|A|$ 的子串的哈希值是否在 $\mathrm{map}/\mathrm{set}$ 里。
- 后缀数组：考虑直接将 AA 和 B 直接拼起来，检查 B 的子串和排名最相近的的 AA 的子串 lcp 是否大于等于 $|A|$ 即可。

时间复杂度 $O(n \log n)$.

02

这题有一个 $O(NK)$ 的做法，就是考虑 $dp_{i,j}$ 表示经过前 i 个机会，获得 j 颗星星的最小代价。

下面是正解部分。

- 正解的思路非常容易&这其实是一个很有意思的问题。
- 也是优化小体积背包的经典做法。

这题具有一个 $O(N\sqrt{K})$ 的做法。

就是首先考虑顺序无关，可以先打乱顺序。

注意到 N 个关卡要获得恰好 K 个星星，那么我们打乱顺序，得到星星的地方是随机并且均匀排布的。（防止被故意构造卡掉）

当选择了前 i 个机会，那么会期望获得 $E = \frac{iK}{N}$ 个星星，而我们只需要在这个期望值附近的一些位置，暴力更新 dp 即可。

- 本题的体积最大值为 $v = 4$.
- 至于这个区域选择的大小，应当至少是 $B = v\sqrt{K}$。
- 我们只需要在 $[E - B, E + B]$ 附近更新 dp 值即可。

所以可以知道复杂度是 $O(N\sqrt{K})$ 的。

假设每个地方只有 $0/1$ 颗星星，而非 $0/1/2/3/4$，下面有一组数据可以供大家参考。

下面的是选取 N, K 以及错误率 ε 所需要的区域大小 B。（$K = \frac{N}{2}$ 是错误率最高的时候）

N	K	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$	$\varepsilon = 10^{-9}$
100	50	11	12	13	14	15	16
1000	500	35	39	42	46	49	51
10000	5000	111	123	134	145	154	163
100000	50000	352	390	426	458	489	517
1000000	500000	1112	1235	1346	1449	1545	1636

这个数据的来源是：https://arxiv.org/pdf/1309.4029.pdf

03

考虑 $\max(a_u, a_v) \times |a_u - a_v|$ 其实是 $\max^2 - \max \times \min$。

由于 $\max \times \min = a_u \times a_v$，这部分是可以简单计算的，答案是：$\left(\sum_{u \in subtree(i)} a_u\right)^2$ 。

剩下的问题是 $\sum_{u,v \in subtree(i)} (\max(a_u, a_v))^2$。

对于子树问题，可以想到按照每一条边 (u, v)（v 是 u 的某个儿子）依次加入。

具体过程为：

- $subtree(u)$ 初始为 $\{u\}$。
- 计算 $subtree(v)$ 和当前 $subtree(u)$ 之间点对的答案。（跨越 u 节点的部分）。
- 把 $subtree(v)$ 子树内的答案直接累加。（不跨越 u 节点的部分）。
- $subtree(u) \leftarrow subtree(u) + subtree(v)$（将 v 的子树加入到 u 中）。

对于第四步容易想到启发式合并/线段树合并。

如果采用启发式合并，复杂度是 $O(n \log^2 n)$ 的。如果采用线段树合并，复杂度是 $O(n \log n)$ 的。当然也可以选择 dsu on tree，复杂度 $O(n \log^2 n)$。

04

线段树分治+可撤销并查集+lazytag。

- 并查集是一个树形结构。
- 线段树分治到叶子结点的时候，对 1 所在的连通块的根节点打上一个 lazytag，表示这一整个子树需要加上一个值。
- 断开 (u, v) 的话，需要将 v 加上 u 的 lazytag。（假设 u 是 v 的祖先）
- 如果后续需要连上 (u, v)，需要将 v 减掉 u 的 lazytag。（因为 v 在之前时刻不属于 u，这是不属于 v 的部分）

05

考虑 n 为偶数，分两种情况，平局和非平局。

平局的概率假设是 p，那么非平局的时候两人互换结果，输赢翻转，所以 A 赢的概率为 $\frac{1-p}{2}$，平局的概率直接用组合数就能表示。

考虑 n 为奇数，分两种情况，前 $\frac{n-1}{2}$ 个字符两人相同和不同。

两人相同则一定是 A 赢，假设概率为 p 两人不同则可以互换前 $\frac{n-1}{2}$ 个字符，输赢翻转，所以 A 赢的概率为 $\frac{1+p}{2}$，p 直接用组合数就能表示。

06

问题等价于我们选三个序列，他们恰好相同的方案数。

考虑 $f[i][j][k]$ 表示现在选了三个一样的序列，第一个序列最后一位为 a_i，第二个为 a_j，第三个为 a_k 的方案数，需要满足 $a_i = a_j = a_k$，转移利用前缀和优化，复杂度为 $O(n^3)$。

07

考虑 3 个点的有向图，要么成环，要么有一个点入度为 2，假设第 i 个点的入度为 d_i，答案为

$$\binom{n}{3}-\sum_{i=1}^{n}\binom{d_i}{2}$$

d_i 利用三维偏序求解。

08

首先每个位都是可以拆开的，而且 0 的方案数为 4，1 的方案数为 12，把每一位的方案数乘一下就可以了。

09

很显然是个数位 DP 题，不过应该细节有点多。

考虑对于一个确定的数字（或者说 $l=r$ 的情况），我们可以设计 $f_{i,j,x}$ 表示扫完了数字串从高到低前 i 位，当前的子序列长度为 j，且最后一个数字符是 x 的合法方案数，这是一个 $O(nm|\Sigma|)$（n 是数字串长度，m 是关系串长度，Σ 是数字字符集 $0\sim 9$）的算法。

如果对于一个区间内的所有数字，首先我们根据套路转换成两个前缀的答案相减。

数位 DP 可以将问题变为 $n|\Sigma|$ 次前面半个数字串确定，后面半个数字串每个字符都在 $0\sim 9$ 中随意填然后合并两半答案的问题。

对于前半，我们只需要继承确定数字的做法 DP 就可以了。

对于后半，我们设计 $g_{b=0/1,i,j,x}$ 表示前面是或不是（靠 b 确定）前导零，从高到低第 i 位之后随意填，子序列从第 j 位开始算，上一个数字符是 x 的合法方案数，这个数位 DP 大概是 $O(nm|\Sigma|^2)$ 的，推荐用记忆化搜索的方式写。

之后合并这两半只需要暴力枚举就可以了，大概也是 $O(nm|\Sigma|^2)$ 的。

由于常数比较小且跑不满，这个做法完全跑的过去。

10

容易发现随机情况下答案极大概率是大数，所以把前 K（K 是一个比较小的数）大拉出来算一下出现次数即可。

具体实现时可以利用类似超级钢琴的做法：开一个堆存区间，每次把最大值最大的区间 $[l,r]$ 取出来，假设最大值为 x，位置为 p，那么这个区间对 x 的出现次数的贡献为 $(p-l+1)\cdot(r-p+1)$，然后再把这个区间分裂成 $[l,p-1]$ 和 $[p+1,r]$ 丢回堆中。

时间复杂度 $O(nK\log n)$，其中 K 为取前几大。

11

首先二分 mex 为 k，问题相当于找一条链覆盖 $0\sim k-1$。

考虑 $0\sim k-1$ 中的某个点权 v，其可以让不经过点权为 v 的链 (x,y) 不合法。

可以发现，这样的 (x,y) 对于特定点权 v，按照 dfs 序排列是若干个矩形的并。

具体而言，对于每个点权 v，我们可以构建一棵虚树，每棵虚树的点集是点权为 v 的点，边是点往其最近的相同点权的祖先连边。那么对于每个点，其每个在原树上的儿子的子树去掉内部虚树上的儿子的子树之间的所有点之间的点对都不合法，这些点在 dfs 序上可以被描述为若干个区间，这些区间作为边的所有矩形就是所有不合法的点对。

如果没有什么特殊限制，矩形的数量是可以到达 $O(n^2)$ 级别的。但是由于题目的特殊限制，对于每个点这若干个区间最多只有 3 个，这样矩形的总数量就是线性的了。

剩下就用扫描线线段树处理即可。

时间复杂度 $O(n \log^2 n)$。

12

考虑把坐标离散化，利用二维前缀和，计算出 g_c 表示被 n 个矩形中的 c 个矩形覆盖的面积和。

那么随机选出覆盖 k 个矩形的面积即为

$$\sum_{i=1}^{n} \frac{1 - \binom{n-i}{k}}{\binom{n}{k}} \times g_i$$