第十八届黑龙江省大学生程序设计竞赛

题解

吉林大学命题组

May 15, 2022

Overview

1. A. Magic Computer
2. F. Folder
3. E. Ethernet
4. I. Club
5. G. Gravity
6. B. Chevonne' s Game
7. K. Turn-based Game
8. J. XOR String
9. C. Compass
10. L. Subxor
11. H. KingZ
12. D. Pandemic

A. Magic Computer

通过观察样例和打表可以发现，$n = 2^{k-1}$。

A. Magic Computer

- $n = 2^{k-1}$ 是可行的：归纳证明，$k = 1$ 的时候 $n_{min} = 1$，这是显然的，当 $k = K$ 时命题成立，我们往证 $k = K + 1$ 时命题成立，先把 2^{K-1} 个 U 盘插上并让他们都有 K 个文件，再每次拔出一个和另外 2^{K-1} 个文件一一配对即可。
- 下证 $n = 2^{k-1}$ 是下界：我们对每个 U 盘定义它的状态值 $P = 2^{L-1}$，其中 L 是当前 U 盘中的文件数且大于 1。特殊地，当 $L = 1$ 时 $P = 0$ 那么我们定义势能函数 $F = \sum_{\text{all USB disks}} P - [\text{the number of USB disks that not on computer and contain more than one file}]$ 在第一次插入 2U 盘后 $F = 1$，那么我们试图证明：在一次拔出 1U 盘，插入 2U 盘过程中，F 不降：
    - 2 个 U 盘开始都是只有一个文件，那么插入后他们的 P 都变成 0.5, 但拔出的时候必有一个文件数大于 1 的 U 盘被拔出，F 不变
    - 一个 U 盘只有一个文件有一个文件，一个有 $m, (1 \leq m)$ 个文件，那么他们 P 都变成 2^m,P 和不变，未插上电脑且文件数大于 1 的 U 盘的数量也不变。
    - 2 个 U 盘的文件数都大于 1，P 和的最大损失也不超过 $0.5 + 0.5 = 1$, 而未插上电脑且文件数大于 1 的 U 盘的数量减少 1，所以 F 不降

A. Magic Computer

故，最后 F 大于等于 1 而最后每个 U 盘的 P 都小于等于 2^{-k+1}，所以至少有 2^{k-1} 个 U 盘。

F. Folder

文件夹显然是一棵树的结构，原题相当于把一个树的其中一个子树切下来，接到另一个节点上，问最少进行几次操作使得树变成一个链。
而每一次操作，显然可以且至多减少一个叶子，故只需要数原始树的叶子结点个数，答案即为叶子结点个数减一。

E. Ethernet

方法一：使用 dfs 统计出分配的方案总数 M，以及这 M 个方案中第 n 个网线被插入第 n 个端口的情况数量 N，最后输出 N/M 即可。
方法二：将方案数统计替换成状态压缩 dp。
方法三：我们定义随机分配的网线为"游离状态"。可以发现，之后每插入一根新的网线，若对应端口没被占用，则"游离状态"的网线数量不变；若对应端口被占用，原来"游离状态"的网线其中一条被固定，新加入的网线变成"游离状态"。因此，"游离状态"的网线数量始终为 m，则加入最后一条网线 n 时，这条网线被插入 n 号端口的概率为 $1/(m+1)$。注意需要特判 $n=m$ 的情况，此时需要输出 $1/m$。

I. Club

所有类型的奖章显然应该平摊给每个社团，则奖章可分为两类，一类分给了 $\lceil \frac{n}{m} \rceil$ 个社团，一类分给了 $\lceil \frac{n}{m} \rceil - 1$ 个社团。设两类奖章的数量分别为 A, B 种，被选到的概率分别为 p, q。则 $A = (n-1)\%m + 1, B = m - A, p = \frac{\lceil \frac{n}{m} \rceil}{n}, q = \frac{\lceil \frac{n}{m} \rceil - 1}{n}$。由此，我们可以进行期望 dp，定义 $dp_{i,j}$ 为从第一类，第二类奖章分别差 i 种，j 种全部选择完的状态开始，还需要进行游戏次数的期望值。则我们有初值 $dp_{0,0} = 0$，可以得到如下方程

$$dp_{i,j} = (1 - pi - qj)dp_{i,j} + pidp_{i-1,j} + qjdp_{i,j-1} + 1$$

进行整理可得转移如下

$$dp_{i,j} = \frac{pi}{pi+qj}dp_{i-1,j} + \frac{qj}{pi+qj}dp_{i,j-1} + \frac{1}{pi+qj}$$

$dp_{A,B}$ 即所求最终结果。在 $A = B$ 时时间复杂度达到最大值，即 $\mathcal{O}(m^2)$。

G. Gravity

设 $v_i = \overrightarrow{OP_i}$，第一天收集的行星集合为 A，第二天收集的行星集合为 B，则题目实际需要我们计算 $\frac{1}{2}\left(\frac{\sum_{i\in A} v_i}{|A|} \times \frac{\sum_{i\in B} v_i}{|B|}\right)$ 的最大值。

可以发现，$\frac{1}{2}\left(\frac{\sum_{i\in A} v_i}{|A|} \times \frac{\sum_{i\in B} v_i}{|B|}\right) = \frac{(\sum_{i\in A} v_i)\times(\sum_{i\in B} v_i)}{2|A||B|} = \frac{((\sum_{i\in A} v_i)+(\sum_{i\in B} v_i))\times(\sum_{i\in B} v_i)}{2|A||B|} = \frac{S\times(\sum_{i\in B} v_i)}{2|A||B|}$。其中 S 为所有向量之和，即 $S = \sum_{i=1}^{n} v_i$。

随后我们对分母使用乘法分配律，上式变为了 $\frac{\sum_{i\in B}(S\times v_i)}{2|A||B|}$。为了求得这个式子的最大值，我们可以把 v_i 根据 $S\times v_i$ 的大小排序。然后枚举集合 B 的大小，选取 $S\times v_i$ 前 $|B|$ 大的数值即可。时间复杂度 $O(n\log n)$。

B. Chevonne’s Game

先考虑没有修改单次询问的情况，如何才能找到最优解。
我们把待询问的串视作由多个极长的 01 交替串所构成，那么答案的一个上界就是这样极长串的段数——因为我们每次随便拿走一段极长的 01 交替串，拿走的次数肯定不大于这个上界。
然后再考虑如何去最小化这个答案。我们发现如果一个极长的 01 交替串的两边分别是类似于 00 与 11 的结构，那么拿走这个串的话，极长串的总段数是会减小 2 的（这个串本身，以及合并出来一个 01/10），其他情况下总段数只会减小 1。
于是我们只需要先拿走所有这样减小 2 的段数。如果我们记 00 这种结构的数量是 S0，11 结构的数量是 S1，那么我们最多可以操作 min(S0,S1) 次，而总的极长串的段数是 S0+S1+1，所以剩下还需要操作 S0+S1+1-min(S0,S1)*2 次，也就是 max(S0,S1)-min(S0,S1)+1 次，所以从操作次数就是 max(S0,S1)+1 次。
对于多次询问有修改的情况，用线段树维护一下即可。

K. Turn-based Game

一句话题意：现有若干场战斗，需要制定策略来最小化盾量损耗。每场战斗有若干回合，直到杀死所有怪兽为止，每一回合先由玩家选择一只怪兽进行一次攻击，再由所有存活怪兽对玩家进行一次攻击。每场战斗，玩家的第 i 次攻击可能得到特殊效果，效果为若杀死一只怪兽即可以立刻再进行一次攻击。

K. Turn-based Game

一次攻击分为杀死怪兽和未杀死怪兽两种情况。经过分析可以发现，未杀死怪兽的攻击需要优先攻击血量大于 1 的怪兽中血量最少的那只，这样可以更早的得到杀死怪兽的机会。
考虑一次战斗的计算。采用动态规划，定义 g[i][j] 表示前 i 次攻击、杀了 j 个怪这一状态下的最小盾量损耗。转移枚举第 i+1 次攻击不杀怪/杀怪（需要判断合法）即可，特别的，有 Seel 加成时本次攻击盾量损耗为 0。
考虑所有战斗如何计算。怪物的攻击造成盾量损耗为初始怪物数量减去存活怪物数量，所以可以把动态规划的结果定义为一个以初始怪物数量为变量的一次函数，考虑变量系数为攻击次数减去 Seel 加成次数，所以可以更改状态定义。
f[i][j][k] 表示前 i 次攻击，杀了 j 个怪，用了 k 次 Seele 加成的最小代价，转移枚举第 i+1 次攻击不杀怪/杀怪（需要判断合法），如果杀怪 Seel 加成需要不改变代价。
对于每次战斗，枚举 k，代入初始怪物数量即可获得具体的最小盾量损耗。

J. XOR Stringn

一句话题意：
现有一个循环字符串，每个位置有一个权值，求其最长子串，满足该子串的所有出现位置（子串出现位置被定义为该次出现中其第一个字符的位置）异或和为 0。

J. XOR Stringn

循环字符串可以通过倍长进行处理。
对倍长后的字符串，建立后缀树。
后缀树的每个节点对应原串的一个子串（根节点到该节点路径构成的串），所有以该子串开头的后缀信息都在该节点子树中，对子树中所有有效后缀的位置权值（倍长串的有效后缀是那些长度大于原串的后缀），判断其异或和是否为 0，对所有合法子串的长度取最大值即可。

C. Compass

考虑先去枚举其中一个 t，不妨设为 t_0。由于一个 t 会带动两层圆环的移动，因此一个 t_0 的枚举量为 $lcm(y_1, y_2)$，这里的 lcm 为最小公倍数。
然后我们需要快速计算出最小的 $t_1 + t_2$，同时满足上面所有条件。设我们枚举的 $t_0 = x$，则 t_1, t_2 需要满足下面的条件：

$$x_0 + t_1 + t_2 \equiv 0 \pmod{y_0}, t_2 = py_1 - (x_1 + x), t_1 = qy_2 - (x_2 + x)$$

将 t_1, t_2 代入第一个式子，可以得到 $py_1 + qy_2 \equiv (x_1 + x) + (x_2 + x) - x_0 \pmod{y_0}$。我们目标是找到最小的 $py_1 + qy_2$。
可以在 $O(y^2)$ 时间复杂度内预处理 $(py_1 + qy_2) \bmod y_0 = w$ 时的最小的 $py_1 + qy_2$。对于枚举到的 $t_0 = x$ 我们去查表即可。

L. Subxor

首先可以对序列进行一次前缀异或处理，将区间问题变成两点问题。
考虑将原序列分成 $\sqrt{n}$ 段。从左到右依次插入数字，对于插入的数字 a_r，计算在每块中最靠左的满足 $w[l] \oplus w[r] <= K$ 的 l，然后将这块的价值对 $r-l$ 取 max。则对于一个询问 $[l, r]$，其答案就是 $[l, r]$ 中间所有块的价值的最大值，以及需要对若干个以 l_i 为左端点的最长子区间长度取最大值。
计算在每块中最靠左的满足 $w[l] \oplus w[r] <= K$ 的 l，可以对于每块建立 *trie* 树，在 trie 树中的每个子树维护下标最小值。计算以 l_i 为左端点的最长子区间，可以对整个数组建立一根 *trie* 树，在 trie 树中的每个子树维护下标最大值。
单次插入需要更改 $O(\sqrt{n})$ 个块，单次询问最多查询 $O(\sqrt{n})$ 个左端点，复杂度为 $O(n\sqrt{n}\log n)$。

H. KingZ

建图，从小到大爆搜答案然后跑上有源汇的上下界可行流即可。
根据等待的回合数，计算部队驻扎增量规则修改真实部队驻扎量，设等待若干回合后的新局势图为 $a_{i,j}$ ，那么按照以下规则建图
将棋盘上每个点拆成三个点，分别为初始态 $s_{i,j}$ ，进攻态 $att_{i,j}$ ，占领态 $occ_{i,j}$。
对于当前状态下已经占领的点 (i,j) ，从源点 S 向 $s_{i,j}$ 连一条流量为 $[0, a_{i,j}-1]$ 的边 $(S, s_{i,j}, 0, a_{i,j}-1)$（后续题解中直接将边写为四元组形式）表示从这个点出发有这么多的部队可供调遣，同时连边 $(S, occ_{i,j}, 0, 1)$ 表示这个点一开始就已经占领。
对于所有的点，连边 $(occ_{i,j}, T, 1, INF)$ 表示所有的点最终都需要占领；
对于已占领点 (i,j) 和未占领点 (x,y) ，连边 $(s_{i,j}, att_{x,y}, 0, tim+dis)$ 表示从 (i,j) 派兵空降前往 (x,y) 且满足 $<\quad + dis$ 的要求。
对于未占领点 (i,j) ，连边 $(att_{i,j}, occ_{i,j}, a_{i,j}+1, INF)$ 表示占领这个点最少要 $a_{i,j}+1$ 的兵力。
那么，对于这个图跑一遍有源汇的上下界可行流，若有解，则说明这个回合数是可以占领所有点的，若无解，则说明这个回合数是不可以占领所有点的。

D. Pandemic

方法一:
定义 $g(n)$ 为连续 n 个寝室拿出 $4n$ 盒盒饭的方案数，由于盒饭共有 m 种且每种数量无限，故 $g(n)$ 的值为从 m 里选 $4n$ 的可重组合，即 $\binom{m+4n-1}{4n}$，定义域为 $[1, K]$ 间的整数
定义 $f(n)$ 为为 1 至 n 寝室发餐的方案数，不难得到如下关系式

$$\begin{cases} f(n) = 1 & n = 0 \\ f(n) = \sum_{i=1}^{\min(n,K)} g(i) f(n-i) & n \geq 1 \end{cases}$$

使用分治 FFT 处理即可。

D. Pandemic

方法二：

考虑构造上述方法一中 $g(n)$ 数组的普通型生成函数 $G(x)=\sum_{n=1}^{K} g(n)x^n$，$f(n)$ 数组的普通型生成函数同理为 $F(x)=\sum_{n=0}^{+\infty} f(n)x^n$，由上述转移式我们发现 $F(x)=1+G(x)+G^2(x)+\cdots+G^n(x)+\cdots$，通过形式幂级数的求和公式我们得到 $F(x)=\dfrac{1}{1-G(x)}$。故处理出 $1-G(x)$ 的前 $n+1$ 项后对其进行多项式求逆即可，最后答案为 $F(x)$ 中 n 次方项的系数。