2019 Nanchang Regional Contest

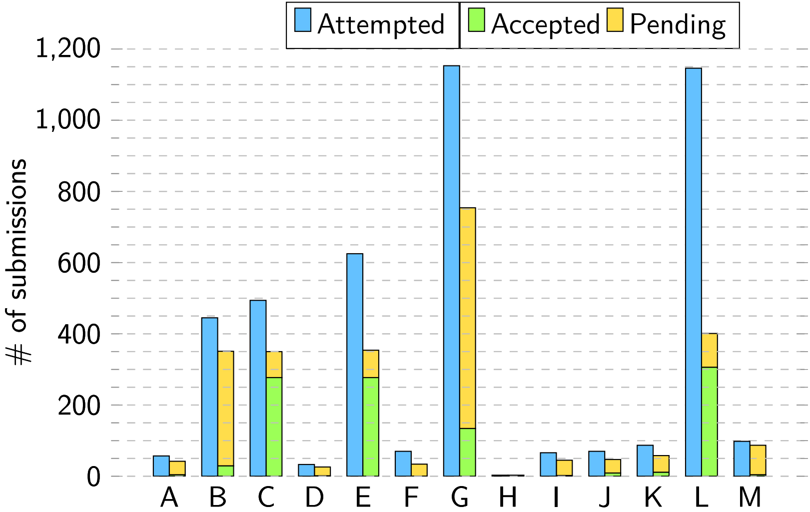
Solutions

Nov. 10th, 2019

Contest Submmary

- 4408 submissions in total;
- 332 teams (all teams) submitted at least once;
- 316 to 332 teams passed at least one problem.

Contest Summary



L. Who is the Champion

题意

N 支球队进行足球比赛，每两支球队均进行一场比赛，胜方积三分，负方积零分，若打平则两队均积一分。按照积分与净胜球进行排名，问最后哪支球队获得冠军，若无法决出冠军则输出“play-offs”。

小模拟，按照题意排序即可，签到题

E. Bob’s Problem

题意

给一个无向图 G，有 n 个点 m 条边，边分为黑边和白边。选不超过 k 条白边使得图联通并且边权和最大。

E. Bob’s Problem

题意

给一个无向图 G, 有 n 个点 m 条边，边分为黑边和白边。选不超过 k 条白边使得图联通并且边权和最大。

因为权值不为负，所以黑边全选，将黑边相连的点用并查集并起来。将白边排序做最大生成树，选出 t 条白边。

如果 $t > k$，则无解。否则从大到小添加没被选入的白边，直到不能选。此时输出答案。

C. And and Pair

根据题意写出答案的表达式：

$$\sum_{i=0}^{n}\sum_{j=0}^{i}[i\&n=i][i\&j=0]$$

$$=1+\sum_{i=1}^{n}[i\&n=i]\sum_{j=0}^{i}[i\&j=0]$$

$$=1+\sum_{i=1}^{n}[i\&n=i]2^{\lfloor\log_2 i\rfloor+1-cnt(i)}$$ 考虑 i 的二进制表示中有的 0

根据题意写出答案的表达式:

$$\sum_{i=0}^{n}\sum_{j=0}^{i}[i\&n=i][i\&j=0]$$

$$=1+\sum_{i=1}^{n}[i\&n=i]\sum_{j=0}^{i}[i\&j=0]$$

$$=1+\sum_{i=1}^{n}[i\&n=i]2^{\lfloor\log_2 i\rfloor+1-cnt(i)}$$ 考虑 i 的二进制表示中有的 0

上式中，$cnt(i)$ 表示 i 的二进制表示中 1 的个数。

接下来，考虑 n 的二进制表示，若将其考虑成集合，i 只会取 n 的子集。考虑 i 的最高有效位是第 k 位，而 n 的第 0 至第 $k-1$ 位有 $c(k)$ 个 1，则答案为：

$$
\begin{aligned}
&1+\sum_{k=0}[2^k\&n=2^k]2^{k+1}\sum_{j=0}^{c(k)}\binom{c(k)}{j}2^{-(1+j)}\\
=&1+\sum_{k=0}[2^k\&n=2^k]2^{k}\sum_{j=0}^{c(k)}\binom{c(k)}{j}(2^{-1})^{j}\\
=&1+\sum_{k=0}[2^k\&n=2^k]2^{k}(1+\frac{1}{2})^{c(k)} \qquad \text{二项式定理}\\
=&1+\sum_{k=0}[2^k\&n=2^k]2^{k}(\frac{3}{2})^{c(k)}
\end{aligned}
$$

因此，我们从低位往高位找 n 的 1，同时维护 2^k 和 $(\frac{3}{2})^{c(k)}$，就可以直接计算答案了，复杂度为 $O(\log n)$。

G. Eating Plan

题意

给一个序列 b，b 由 1 到 n 的阶乘构成，m 次询问你子串中子串长度最短的大于等于 k 的子串长度是多少；

G. Eating Plan

题意

给一个序列 b，b 由 1 到 n 的阶乘构成，m 次询问你子串中子串长度最短的大于等于 k 的子串长度是多少；

这是个华点题目，那个 mod 是唬人玩的，它其实是一个合数，$998857459 = 461 \times 773 \times 2803$；

然后你就会发现 b 中不为 0 的数最多只有 2802 个，你可以暴力枚举这 2802 个数构成的子串和，排序，再对询问排序一下，双指针扫一遍即可得出答案 s

B. A Funny Bipartite Graph

题意

给定二分图，左右各 $n(n \leqslant 18)$ 个节点，只有当 $i \leqslant j$ 时 L_i 和 R_j 才有可能有边，每个左节点度数为 $1-3$。
你选出一些边，形成一个子图，需要覆盖所有右节点，有矩阵 A，$A_{ij}=1$ 表示新图不能同时覆盖左节点 L_i 和 L_j。
定义费用：如果左节点 L_i 不在子图中，则费用为 0，否则（它在子图中）费用为 $M_i^{d_i}$ d_i 为它在子图的度数
需要你求最小费用，无解则为 -1 。

B. A Funny Bipartite Graph

题意

给定二分图，左右各 $n(n \leqslant 18)$ 个节点，只有当 $i \leqslant j$ 时 L_i 和 R_j 才有可能有边，每个左节点度数为 $1-3$。
你选出一些边，形成一个子图，需要覆盖所有右节点，有矩阵 A, $A_{ij}=1$ 表示新图不能同时覆盖左节点 L_i 和 L_j。
定义费用：如果左节点 L_i 不在子图中，则费用为 0，否则（它在子图中）费用为 $M_i^{d_i}$ d_i 为它在子图的度数
需要你求最小费用，无解则为 -1 。

如果不考虑复杂度，我们可以有以下做法：

B. A Funny Bipartite Graph

题意

给定二分图，左右各 $n(n \leqslant 18)$ 个节点，只有当 $i \leqslant j$ 时 L_i 和 R_j 才有可能有边，每个左节点度数为 $1-3$。
你选出一些边，形成一个子图，需要覆盖所有右节点，有矩阵 A，$A_{ij}=1$ 表示新图不能同时覆盖左节点 L_i 和 L_j。
定义费用：如果左节点 L_i 不在子图中，则费用为 0，否则（它在子图中）费用为 $M_i^{d_i}$ d_i 为它在子图的度数
需要你求最小费用，无解则为 -1 。

如果不考虑复杂度，我们可以有以下做法：

我们记录 dp[L][R][i] 是已经满足下列条件的，要完成覆盖右节点任务还需要的最少费用

(1) 已经决定了左边第 $1, 2, \ldots, i$ 个节点是否在子图中，L 表示哪些左节点在子图中

(2) R 表示右边哪些节点还没被覆盖

B. A Funny Bipartite Graph

我们只需要求出 `dp[0][所有右节点][0]` **即可**

如果直接状压，需要 $O(2^n \times 2^n \times n)$ **空间，显然太大**

B. A Funny Bipartite Graph

我们只需要求出 dp[0][所有右节点][0] 即可

如果直接状压，需要 $O(2^n \times 2^n \times n)$ 空间，显然太大

但是，可以发现，如果左边 $1-i$ 已经决定了话，右边 $1-i$ 如果有节点 x 还没被覆盖，显然左边 $i+1$ 连的边覆盖不到 x 了，（只有当 $i \leqslant j$ 时 L_i 和 R_j 才有可能有边），这是一个无效状态。这就说明 R 只需要记录（$i+1$ 到 n）哪些点未被覆盖。

B. A Funny Bipartite Graph

我们只需要求出 dp[0][所有右节点][0] 即可

如果直接状压，需要 $O(2^n \times 2^n \times n)$ 空间，显然太大

但是，可以发现，如果左边 $1-i$ 已经决定了话，右边 $1-i$ 如果有节点 x 还没被覆盖，显然左边 $i+1$ 连的边覆盖不到 x 了，(只有当 $i \leqslant j$ 时 L_i 和 R_j 才有可能有边)，这是一个无效状态。这就说明 R 只需要记录（$i+1$ 到 n）哪些点未被覆盖。

而如果已经决定左节点 $1-i$ 是否在子图，那显然左节点 $i+1$ 到 n 未在子图，L 只需要记录左边 $1-i$ 哪些在子图，L 和 R 一共 n 位！我们只需要维护 dp[RL][i] 即可，其中 LR 低 i 位表示 L，LR 高 $n-i$ 位表示 R。转移就很显然了.

K. Tree

可以对 $1-n$ 每个权值建一个平衡树，权值为节点所对应的深度，

K. Tree

可以对 $1-n$ 每个权值建一个平衡树，权值为节点所对应的深度，使用 dsu on tree，在 dfs 的过程中枚举 lca

K. Tree

可以对 $1-n$ 每个权值建一个平衡树，权值为节点所对应的深度，使用 dsu on tree，在 dfs 的过程中枚举 lca，保留重儿子，暴力枚举其余子节点，可以得到所对应的另一个节点的权值

K. Tree

可以对 $1-n$ 每个权值建一个平衡树，权值为节点所对应的深度，使用 dsu on tree，在 dfs 的过程中枚举 lca，保留重儿子，暴力枚举其余子节点，可以得到所对应的另一个节点的权值，在相应的平衡树中查询满足深度要求的节点数有多少即可。

J. Summon

题意

在一个长度为 n 的环上填 $0,1,2,3$，有 m 个长度为 4 的连续的顺时针的子段不能出现。在旋转之后相同的方案算是同一种。问最终有多少种不同的填法。

题意

在一个长度为 n 的环上填 $0, 1, 2, 3$，有 m 个长度为 4 的连续的顺时针的子段不能出现。在旋转之后相同的方案算是同一种。问最终有多少种不同的填法。

用 $F(L)$ 表示在给定条件下不考虑同构的条件时长度为 L 的环的染色方案数，则根据 Polya 原理，可以得到最终答案应为：

$$\frac{\sum_{d|n} \phi(\frac{n}{d})F(d)}{n}$$

下面考虑如何求 $F(L)$。

如果是一条链的情况，那么用 $g[i][s_1][s_2][s_3]$ 表示只考虑前 $i+2$ 个位置，在从第 i 位开始三个位置颜色分别是 s_1, s_2, s_3 时的方案数是多少。对于 $g[i-1][s_1][s_2][s_3]$ 和 $g[i][s_2][s_3][s_4]$，只要 $s_1s_2s_3s_4$ 不是不合法序列就可以转移。最终答案就是 $F(L) = \sum_{0 \leqslant s_1, s_2, s_3 \leqslant 3} g[L-2][s_1][s_2][s_3]$。

下面考虑如何求 $F(L)$。

如果是一条链的情况，那么用 $g[i][s_1][s_2][s_3]$ 表示只考虑前 $i+2$ 个位置，在从第 i 位开始三个位置颜色分别是 s_1, s_2, s_3 时的方案数是多少。对于 $g[i-1][s_1][s_2][s_3]$ 和 $g[i][s_2][s_3][s_4]$，只要 $s_1s_2s_3s_4$ 不是不合法序列就可以转移。最终答案就是 $F(L) = \sum_{0 \leqslant s_1, s_2, s_3 \leqslant 3} g[L-2][s_1][s_2][s_3]$。

将 $s_1s_2s_3$ 视为三位四进制数，那么显然可以构造出一个 $4^3 * 4^3$ 的转移矩阵 A 进行矩阵乘法，于是用矩阵快速幂，最后优化复杂度为 $64^3 \log_2 L$。

考虑将这个做法扩展到环上。链的做法经过了 $L-2$ 次转移，而环要求首位相接，那么我们考虑枚举初始状态为 $s_1s_2s_3$，这个状态经过 L 次变化之后应该还要是 $s_1s_2s_3$。于是将矩阵自乘 L 次，得到的矩阵对角线上元素之和就是答案。这时的时间复杂度为 $O(64^3 \log_2 L \times \text{因子个数})$，可能不够快。

J. Summon

一个更优秀的做法是，由于需要多次计算 A 的次幂，可以设定一个 $m(sqrt(n) \leqslant m)$，分别求出 A^i 和 $A^{m \times i}(0 \leqslant i \leqslant m)$，这样每次计算 A 的次幂的时候，做一次矩阵乘法 $(A^m)^{L/m} \times A^{L\%m}$ 即可。时间复杂度为 $O(64^3 \times sqrt(n))$

注意特殊处理 $F(1), F(2)$

A. 9102

把询问离线下来建树后 dfs，

A. 9102

把询问离线下来建树后 dfs，每个节点处理操作，操作用可撤回并查集实现，

A. 9102

把询问离线下来建树后 dfs，每个节点处理操作，操作用可撤回并查集实现，时间复杂度是 $O(m \log n)$

M. XOR Sum

首先有

$$\oplus_{i=1}^{n} i = \begin{cases} n, & n \bmod 4 = 0; \\ 1, & n \bmod 4 = 1; \\ n+1, & n \bmod 4 = 2; \\ 0, & n \bmod 4 = 3. \end{cases}$$

M. XOR Sum

于是有

$$\sum_{k=1}^{t}\sum_{i=1}^{n} f(i,k) = \sum_{k=1}^{t}\sum_{i=1}^{n} \left[i^k \bmod 4 = 0 \text{ or } 2\right] i^k + \sum_{k=1}^{t}\sum_{i=1}^{n} \left[i^k \bmod 4 = 1 \text{ or } 2\right]$$

M. XOR Sum

分析 k 和 i 的奇偶性可以简单计算出后一项，前一项则等于

$$\sum_{k=1}^{t} 2^k \sum_{i=1}^{m} i^k$$

其中 $m = \lfloor \frac{n}{2} \rfloor$，这是一个关于 m 的 $t+1$ 次多项式，使用拉格朗日插值公式代入 $t+2$ 个点值进行单点求值即可。

题意

给一棵根为 1 的树，点带权，边上是二进制与/或/异或的一种。定义一条路径 $u \to v$ 的权值是按顺序从 u 走到 v，点权通过边上的运算得到的值。有 q 次询问，每次询问给出 d, u，表示将第 i 个结点的权值修改为 $a_i + i \times d$，并计算对于所有满足 u 是 v 的祖先的路径 $u \to v$ 的权值的或/与/异或值

D. Bitwise Tree

f[i][j][k] 表示对于每一位

0 进去变成 0 的有没有，0 进去变成 1 的有没有，1 进去变成 0 的有没有，1 进去变成 1 的有没有

利用子节点生成父节点的 f[i][j][k]，用 h[j][k] 表示由子节点的 f 相或所得到的值

D. Bitwise Tree

连接当前节点的符号为 | 时:

$$f[i][0][k]| = (({\sim}a[i])\&h[0][k])|(a[i]\&h[1][k])$$

$$f[i][1][k]| = (({\sim}a[i])\&h[1][k])|(a[i]\&h[1][k])$$

连接当前节点的符号为 & 时:

$$f[i][0][k]| = (({\sim}a[i])\&h[0][k])|(a[i]\&h[0][k])$$

$$f[i][1][k]| = (({\sim}a[i])\&h[0][k])|(a[i]\&h[1][k])$$

连接当前节点的符号为ˆ时，思路类似，只是 $f[i][j][k]$ 表示奇偶性:

$$f[i][0][k]| = (({\sim}a[i])\&h[0][k]) \text{ \^{} } (a[i]\&h[1][k])$$

$$f[i][1][k]| = (({\sim}a[i])\&h[1][k]) \text{ \^{} } (a[i]\&h[0][k])$$

这样每个节点的 f 将与上它所有子节点的 f 值

非叶子节点的答案即为 $((\tilde{}a[x])\&f[x][0][1])|(a[x]\&f[x][1][1])$

I. Resistance

大模拟，略

F. Dynamic Suffix Array

题意

维护一个字符串 S，一开始 S 为空，支持两种操作：

1. S 末尾新增一个小写字符 c, c 在'a' 到'z' 等概率随机产生
2. 给定 k，问 S 那么多后缀里，k 这个后缀排第几小

题意

维护一个字符串 S，一开始 S 为空，支持两种操作：

1. S 末尾新增一个小写字符 c，c 在'a' 到'z' 等概率随机产生
2. 给定 k，问 S 那么多后缀里，k 这个后缀排第几小

乱搞题。

仅考虑前 4 位求排名，前 4 位相同的部分只有几个串，可以暴力算，可以转成 26 进制用 BIT 维护。

H. Powers of Two

考虑一个很小的区间 $[0, \epsilon]$，对于任意区间内的位置 x，我们希望找到最小的<u>**步长**</u> k 满足 $\{x + k\log_{10} 2\}$ 依然在这个区间内。

不难发现存在最小的 k_1 使得 $0 \leqslant \{k_1 \log_{10} 2\} \leqslant \epsilon$。于是记 $\epsilon_1 = \epsilon - \{k_1 \log_{10} 2\}$，则 k_1 作为步长可以满足所有的 $x \in [0, \epsilon_1]$。

类似的，存在最小的 k_2 使得 $1 - \epsilon \leqslant \{k_2 \log_{10} 2\} < 1$。记 $\epsilon_2 = 1 - \{k_2 \log_{10} 2\}$，则 k_2 作为步长可以满足所有的 $x \in [\epsilon_2, \epsilon]$。

最后，若 $0 \leqslant \epsilon_1 < \epsilon_2 \leqslant \epsilon$，对于 $x \in [\epsilon_1, \epsilon_2]$，我们发现最小的步长恰好就是 $k_1 + k_2$。

所以：最小的步长最多只有三种可能。

对于更小的 ϵ，不难发现新的步长一定是刚才求出来的较大的步长 $\{k_1, k_2, k_1 + k_2\}$ 的整系数线性组合，于是可以迭代求出来对于任意小的 ϵ 的三个步长。其实进一步分析会发现，他们还在一定程度上对应了 $\log_{10} 2$ 的连分数。

回到原题，我们取 $\epsilon = \log_{10}(p + 1) - \log_{10}(p)$。再用一些方法找到第一组解（或者构造任意一组解，然后利用步长反推出第一组解），然后利用步长去寻找第 k 组解即可。