

SYSU 校赛题解

WHU ICPC 集训队

2023.3.9

A. Forest

题意

一个 $n (\geq 6)$ 个点无重边无自环的无向连通图，有 $k (0 \leq k < n)$ 个三度点，其余点度小于 3，构造两个满足如上要求的图，使得边数最多和最少。

A. Forest

题意

一个 $n (\geq 6)$ 个点无重边无自环的无向连通图，有 $k (0 \leq k < n)$ 个三度点，其余点度小于 3，构造两个满足如上要求的图，使得边数最多和最少。

题解

设有 a 个 1 度点，则有 $n - k - a$ 个 2 度点，总边数为

$$(3k + 2n - 2k - 2a + a)/2 = (k - a)/2 + n$$

接下来考虑连通性，因为删去 2 度点不影响，所以讨论的时候可以忽略。
下面给出一个参考做法：

题解

如果希望边数最多，则 1 度点尽可能需要少。

$k = 0$ 时，2 度点连一个环即可。

$k = 1$ 时，唯一的 3 度点与两个 2 度点连成一个环后，再补一个 1 度点在 3 度点上，接着补余出来的 2 度点即可。

$k = 2$ 时，2 度点连环后，随便连一对不相连的点。

$k = 3$ 时，去掉一个 1 度点后即上一个情况。

余下情况，三度点连成环后两两连边凑度数，如果是奇数补 1 个一度点即可，余下补 2 度。

题解

如果希望边数最少，在 1 度点尽可能需要多。

$k = 0$ 时，连一条链。

$k * 2 + 2 \leq n$ 时，3 度点连出一条链后，全接 1 度点有 $k + 2$ 个，可以发现是树的情况，2 度点随便补补即可。

否则三度点先连一个环，余下点尽可能选 1 度点（要保证 1 3 两类点同奇偶），1 度不够就 3 度两两相连。

根据实现的不同可能需要讨论 k 奇偶性等问题。

solution1.min25 筛

令

$$x = \prod p_i^{a_i} (p_i \in \text{prime})$$

则令

$$f(x) = \prod (2a_i + 1)$$

$$\text{ans} = \sum_{i=1}^n f(i)$$

直接 min25 筛处理

solution2. 杜教筛

$$\text{ans} = \sum_{p=1}^n \sum_{q=1}^{n/p} [\gcd(p, q) = 1] \frac{n}{pq} \quad (1)$$

$$= \sum_{T=1}^n \frac{n}{T} \sum_{d|T} [\gcd(d, \frac{T}{d}) = 1] \quad (2)$$

$$= \sum_{T=1}^n \frac{n}{T} \sum_{d|T} \mu^2(d) \quad (3)$$

$$= \sum_{d=1}^n \mu^2(d) \sum_{i=1}^{\frac{n}{d}} \frac{n}{id} \quad (4)$$

solution2. 杜教筛

考虑如何计算

$$f(n) = \sum_{i=1}^n \mu^2(i) \quad (5)$$

$$g(n) = \sum_{i=1}^n \frac{n}{i} \quad (6)$$

都采用大于 B 时候根号暴力，小于 B 时候预处理即可，复杂度分析类似杜教筛，由于 g 的预处理是 $n \log n$ 的， B 可以略开大一点，一个很松的复杂度上界是 $O(n^{\frac{2}{3}} \sqrt{\log n})$ ，但实际很快。

solution2. 杜教筛

这里提一下 μ^2 的根号暴力计算：

$$\sum_{i=1}^n \mu^2(i) = \sum_{i=1}^{\sqrt{n}} \frac{n}{i^2} \mu(i)$$

两种做法都可以通过本题。

C. Catch-Hamster

题解

可以证明，我们的目标其实就是使得任意两个叶子结点之间的路径上至少含有一个捕鼠器。

钦定一个非叶子结点为树根，跑一遍 DFS。如果当前结点 v 为叶子结点，那么令 $a[v] = 1$ ；否则，首先令 $a[v] = \sum_u a[u]$ ，如果发现 $a[v] > 1$ 那么在当前结点处放置捕鼠器并修改 $a[v] = 0$ ，这样我们就保证了任意两个叶子结点之间的路径上至少含有一个捕鼠器，同时也避免了在不必要的结点处放置多余的捕鼠器。其正确性显而易见：考虑到如果我们在某个 $a[v] > 1$ 的结点处没有放置捕鼠器，那么仓鼠显然可以在当前子树内的两个叶子结点之间反复横跳，因而既是必要的也是充分的。

题解

首先有一个暴力的费用流做法：

记代表咖啡的点为 x_i ，代表券的点为 y_i ，并建立一个临时汇点。从源点向所有 x_i 连流量为 1，费用为 a_i 的边，从所有 y_i 向临时汇点连流量为 1，费用为 w_i 的边。从所有 x_i 向临时汇点连流量为 1，费用为 0 的边（代表不用优惠券）。对于第 i 优惠券，从 x_1 到 x_{r_i} 的每个点都向 y_i 连流量为 1，费用为 0 的边。最后从临时汇点向汇点连流量为 k ，费用为 0 的边。

题解

记买 i 杯咖啡的最小费用为 $f(i)$ ，可以证明 $f(i)$ 是下凸的，考虑 wqs 二分：二分一个权值，将每杯咖啡的价格减去这个值，问题变为买任意杯咖啡使得费用最小，判断是否买了至少 k 杯。

对于第 i 张优惠券，有两种使用它的情况：

1. 与还未被减免的咖啡 j 匹配，对答案的贡献为 $a_j - w_i$ 。
2. 替换已经匹配的一张券 j ，对答案的贡献为 $w_j - w_i$ 。

当对答案的贡献 ≤ 0 时，这张券会被使用。

题解

可以用堆维护这个过程。将券按 r_i 从小到大 (r_i 相同则 w_i 从大到小) 排序, 维护一个小根堆, 在尝试匹配第 i 张券时, 对于所有 $j \leq r_i$ 且尚未入过堆的咖啡, 将 a_i 入堆, 然后取出堆顶 c , 如果 $c \leq w_i$ (此时对答案贡献 ≤ 0), 则答案加上 $c - w_i$, 并将 w_i 入堆, 同时如果满足情况 1, 则将购买的咖啡数量 $+1$ 。

当券全部尝试匹配过一次之后, 购买剩下的所有价格 ≤ 0 的咖啡, 然后判断最终购买的咖啡数量与 k 的关系即可。

这个过程也可以用模拟费用流来理解。

复杂度 $O(n \log n \log w)$ 。

E. A Counting Problem

题意

求 $[0, 10^n)$ 内 3×2^a 的倍数的个数。

题解

解法一：推式子。

题目要求的其实就是： $\lceil \frac{10^n}{3 \times 2^a} \rceil$ 。

注意题干中有一个非常重要的条件： $a \leq n$ 。因此我们可以将上式进一步化为： $\lceil \frac{5^a 10^{n-a}}{3} \rceil$ 。

我们记 $r = 3 - (5^a 10^{n-a} \bmod 3)$ ，由向上取整的定义可得：

$$\text{ans} = \lceil \frac{5^a 10^{n-a}}{3} \rceil \equiv \frac{5^a 10^{n-a} + r}{3} \bmod 998244353。$$

通过费马小定理和快速幂即可求出 ans 。每次求解的复杂度为 $O(\log n)$ 。

E. A Counting Problem

题解

解法二：矩阵加速数位 dp。

虽然解法二被解法一完爆，但出题人觉得它很有意思，所以还是放上来了。我们知道，一个十进制数如果是 3 的倍数，那么它的各个数位相加的和模 3 一定等于 0。或者说，十进制数的数位相加和模 3 的值与这个数本身模 3 的值相等。

要求 3×2^a 的倍数，就是要求既是 3 的倍数（下称条件一）又是 2^a 的倍数（下称条件二）。

E. A Counting Problem

题解

我们不妨把这个 n 位的十进制数拆成两部分两部分：前（低） a 位和后（高） $n - a$ 位。

因为 10^a 是 2^a 的倍数，所以后 $n - a$ 位无论怎么变，都不会对条件二产生任何的影响。

对于前 a 位，每一个满足条件二的数都可以看成一个 a 位五进制数乘上 2^a 。因此，将前 a 位看成五进制的，就可以在计数过程中无视条件二，只统计对条件一的贡献。

E. A Counting Problem

题解

我们记 $f_{i,j}$ 表示当前 dp 到第 i 位，模 3 等于 j 的数的数量。

对于前 a 位，我们将其看成五进制数，考虑类似秦九韶的方式。模 3 下，原数乘以 5 后，2 会变成 1，1 会变成 2，再考虑新“个”位的 5 种情况 (0, 1, 2, 3, 4)。不难得出下面的状态转移方程：

$$f_{i,0} = 2f_{i-1,0} + 2f_{i-1,1} + f_{i-1,2}$$

$$f_{i,1} = 2f_{i-1,0} + f_{i-1,1} + 2f_{i-1,2}$$

$$f_{i,2} = f_{i-1,0} + 2f_{i-1,1} + 2f_{i-1,2}$$

显然，上述 dp 过程可以用矩阵加速。需要注意的是在算完前 a 位后，别忘了补上 2^a 的影响：当 $2^a \equiv 2 \pmod 3$ 时，应该交换 $f_{a-1,1}$ 和 $f_{a-1,2}$ 。

而后面的 $n - a$ 位，则按十进制的方式，直接考虑数位之和对模 3 的贡献即可。此处不再列出详细的 dp 方程。

单次询问复杂度为： $O(3^3 \log n)$ 。

F. Fighting in Group

题意

n 个人，编号 1 到 n 。每个人都想和与自己编号不相邻的所有人“切磋”至少一遍。且他们都不希望与自己编号相邻的人“切磋”。

每一轮时，你可以安排两队人相互切磋。对于 a 和 b ，只要某一轮中他们都在场且处在不同的队伍里，就认为 a 和 b 彼此切磋过。

问你至少要安排多少轮才能满足这些人的切磋要求。输出方案。

引子

题意可能比较绕，出题人为了避免歧义在题面上给出了不少的例子，导致题面看起来很长，希望没有对选手的心态产生影响（误）。

本题灵感来源于“GXOI/GZOI2019 旅行者”。我们不妨先来考虑这样一个问题：

n 个点，每次选择两个不相交点集，生成两个点集间的还未生成的所有边，问至少需要多少次才能生成一个完全图。

F. Fighting in Group

引子

考虑分治，比如现在 n 个点内的边还没生成。我们知道把某个点放着不选是没有意义的，所以我们记选择的两个点集分别有 a 个点和 $n - a$ 个点。生成了这两个点集间的边之后，我们还需要解决的就是规模为 a 和 $n - a$ 的两个子问题。而显然的，这两个子问题可以同时进行。

我们记答案为 $f(n)$ ，于是有：
$$f(n) = \min_{a=1}^{n-1} (\max(f(a), f(n-a)) + 1)。$$

显然， $f(n)$ 是不降的（数量越多需要的轮次自然越多）。所以有：

$f(n) = f(\lceil \frac{n}{2} \rceil) + 1$ 。边界为 $f(1) = 0$ 。

而现在，我们给这个问题引入了一个新的要求：不能把编号相邻的点连起来。又该怎么做呢？

F. Fighting in Group

题解

同样的，我们的构造思路依然是分治。

不难观察到，加入这个限制之后，最大的区别在于：点不能全上了。

假设我们现在有连续的 n 个点待处理。易证，把它分成两段，中间用一个不上的点隔开是最优的。我们不妨认为左边的段有 a 个点，右边的段有 b 个点，且 $n = a + b + 1$ 。

那么，本轮过后，我们还需要处理的有：

1. 左边 a 个点内部的边和右边 b 个点内部的边。
2. 中间这个点与其它点的边。

而第二点又可以拆成：中间这个点与左边点的边、中间这个点与右边点的边。

F. Fighting in Group

题解

于是，我们又可以将待处理的子问题合并为：

1. 左边 a 个点内部的边和中间这个点与左边点的边。
2. 右边 b 个点内部的边和中间这个点与右边点的边。

也就是左边 $a + 1$ 个点内部的边和右边 $b + 1$ 个点内部的边。

同样的，这两个子问题也可以同时进行。只是在构造方案的时候需要注意，相邻两个子问题中共用了同一个点的段要放在同一个集合里。

于是，我们又可以列出一条式子：

$f(n) = \min_{i=1}^n (\max(f(i), f(n - i + 1)) + 1)$ 。边界是 $f(1) = f(2) = 0$ 。

同样的， $f(n)$ 不降。于是有： $f(n) = f(\lceil \frac{n+1}{2} \rceil) + 1$ 。

因此，分治时，每个 n 的问题划分为 $\lceil \frac{n+1}{2} \rceil$ 和 $\lfloor \frac{n+1}{2} \rfloor$ 的子问题即可。

F. Fighting in Group

例子

上面讲的可能比较抽象，这里举一个 $n = 9$ 的例子吧：

第一轮：待解决 $[1, 9]$ 。

方案： $\{1, 2, 3, 4\}$ vs $\{6, 7, 8, 9\}$ ($\{5\}$)。

第二轮：待解决 $[1, 5], [5, 9]$ 。

方案： $\{4, 5, 6\}$ vs $\{1, 2, 8, 9\}$ ($\{3, 7\}$)。

第三轮：待解决 $[1, 3], [3, 5], [5, 7], [7, 9]$ 。

方案： $\{1, 5, 9\}$ vs $\{3, 7\}$ ($\{2, 4, 6, 8\}$)。

分治结束，构造完成，共 3 轮。

题解

对抗搜索，直接令 $f[x][y][0/1]$ 表示手牌剩余分别为 x, y ，现在是 Alice/Bob 走的最大收益，收益定为 Alice 与 Bob 的得分之差，如果最后初始状态的 f 值大于等于零就可以先手必胜。

状态数是 $2 \sum_{i=1}^{13} (C_n^i)^2$ ，算下来是可以接受的数目。

运用 Alpha-Beta 剪枝优化这个对抗搜索，实际运行就可以轻松通过，不加剪枝可能会被卡。

题解

用 μ 函数优化 DP, 因为

$$[\gcd(a_i, a_{i+1}) = 1] = \sum_{d|a_i, a_{i+1}} \mu(d)$$

所以令 $f[i][j]$ 表示到了 i 位置填 j 的方案数

先求

$$g(d) = \sum_{d|x} f[i][x] \mu(d)$$

再令

$$f[i+1][j] = \sum_{d|j} g(d)$$

即可在总复杂度 $O(nm \log n)$ 的时间复杂度内解决本题。

I. Brackets

题意

给一个带'?'的括号序列，要把'?'替换成'('或')'，定义价值为关于合法括号的长度的一次函数，求序列若干个不交子串的最大价值和。

解法

令 dp_i 表示 $1 \sim i$ 的最大价值和， i 为右端点时，把合法的左端点找出来，并选择其中最大值。

考虑如何快速判断区间 $[l, r]$ 是否为一个合法括号序列。

一个较为常见的套路是把括号转换成 1 与 -1 之类的，然后考虑前缀与后缀和需要什么限制。

I. Brackets

解法

具体的，令 '(' 为 -1，')' 为 1，若枚举右端点 r ，左端点 l 存在两个限制：

1. 把 l 到 r 之间所有的？当成 +1，不存在 k (区间内) 使得 k 到 r 的和小于 0。观察发现，左端点范围为一个区间，可以通过二分等方法求出。
2. 把 l 到 r 之间所有的？当成 -1， l 到 r 的和是一个后缀最小值，即不存在 k (区间内) 使得 k 到 r 的和小于 l 到 r ，为单调栈上的点。

发现区间长度变长，代价增加是线性的，因此可以在线段树上支持若干区间同时增加相应长度。

具体的，枚举右端点，在线段树上通过插入删除维护限制 2 上的点，扩展区间长度（右端点右移）为区间加，限制 1 的条件为查询区间。

根据实现不同可能的细节也有一定差别。

解法

由于题目中的查询操作只与数列中每一个数字的奇偶性有关，所以如果 A_i 为偶数，我们可以将其视为 0，奇数则视为 1。

首先我们先考虑题目的一个简单版本：询问整个序列的答案（即 $X = 1, Y = n$ 的情况）。

这一部分我们可以使用容斥原理减去不合法的情况。具体地，若我们把序列中连续的 0 或连续的 1 压缩成一段，则若 $[L, R]$ 位于同一段，则区间 $[L, R]$ 为不合法的，反之则合法。具体做法我们可以记 w_i 表示第 i 段的数字个数，例如对于序列 01110011，其对应的 $w_i = [1, 3, 2, 2]$ 。则整个序列的答案可以表示成 $\binom{n}{2} - \sum \binom{w_i}{2}$
时间复杂度为 $O(n)$ 。

解法

对于多组询问区间查询，我们可以使用类似前缀和的方式预处理出前缀不合法的子区间数，不妨用 s_i 表示（即 s_i 区间 $[1, i]$ 中不合法的子区间个数） $s_i = s_{i-1} + \text{cnt}_i$ ，其中 cnt_i 表示 A_i 及其左侧中连续的 0/1 的个数。求解 s_i 的时间复杂度为 $O(n)$

与此同时，我们记录 R_i 表示为满足 $\forall t \in [i, R_i], A_i = A_t$ 条件下的最大值（即 A_i 所在段的最右端）。求解 R_i 可以通过将 A 数组倒序扫描一遍的方法在 $O(n)$ 的复杂度下完成。

解法

对于一组询问 $[X, Y]$ ，答案可以通过如下的形式表示：

若 X 和 Y 在同一段，则答案为 0。

若 X 和 Y 不在同一段，我们可以先求出不合法的子区间个数，对于不合法的子区间数可分为 $[X, R_X]$ 和 $[R_X + 1, Y]$ 两部分。

对于区间 $[X, R_X]$ ，任意子区间均为不合法的子区间，数量为 $\text{sum}_1 = \binom{R_X - X + 1}{2}$ ；对于区间 $[R_X + 1, Y]$ ，不合法子区间数量可通过类似前缀和的方式求出 $\text{sum}_2 = S_i - S_{R_X}$ 。

因此，总的合法子区间数可以表示成： $\binom{Y - X + 1}{2} - \text{sum}_1 - \text{sum}_2$ 。

总时间复杂度为 $O(n)$ 。

题意

定义一个长 n 的排列 p 价值为 $\sum_{i=1}^n p_i \oplus i$, 其中 $\oplus$ 表示异或。
求价值最大的长度为 n 的排列。

解法

由异或的性质可以得到，任何一个长度大于 2 的环是不优的。
接着构造尽可能不浪费的方案，如果 $n = 2^k - 1$ ，匹配方法比较显然，考虑从这里入手。
我们一定会尽可能令最高位 1 不浪费，所以最高位 1 的内部显然不去匹配，为了让其他位也减少浪费，按 $2^k - 1, 2^k; 2^k - 2, 2^k + 1; 2^k - 3, 2^k + 2 \dots$ 匹配再进入子问题即可。

题意

本题给出一个初始字符串，一个由 n 个字符串构成的字典，开始时，我们的字符串在字典的某个位置，题目定义了三种魔法操作，分别是将初始字符串两边删除一个字符，增加一个字符，以及从字典的某一个字符串跳跃到另一个字符串，并给出了各种操作所需的花费。

要求将初始字符串变化成字典上另一个字符串，并跳跃到目标字符串所在的位置。题目要求我们给出最小的花费。

解法

不难想到，定义两个不同字符串之间的距离为魔法操作的花费，然后将每个子回文串抽象成一个点，我们就可以将问题转化为图上的最短路径问题。

由于字符串全程都是回文串，而一个字符串中包含的本质不同的子回文串数量是 $O(n)$ 级别的，所以图中的点和边也是 $O(n)$ 级别的。

而操作数并不多，最后的图边数也不会太大，可以用任何一种较快的单源最短路算法解决（没卡 spfa）。

解法

难点在于建图。

建图的关键在于找出字典中的所有本质不同的字符串，然后将它们映射到图上。这里有多种方法解决。

想到这里，我们发现最终建好的图像极了一个自动机，没错！我们可以直接使用回文自动机，然后把建出来的回文树作为最终的图，跑单源最短路。当然，只要能处理出所有本质不同的字符串，然后把它们映射出来建图，就可以解决这道问题，比如 manacher 算法也可以解决这个问题。本题给出了 manacher 和 PAM 两种算法的 STD。时间复杂度均为 $O(n \log n)$ 。并且 manacher 的效率和代码难度均低于 PAM。