第二十三届电子科技大学程序设计竞赛（初赛）题解

A. 炼金术士

一句话题解：拉姆齐定理+图兰定理

首先划分两次二分图的思想是错误的，因为二分图保证不包含奇数环而不仅仅是三元环，实际构造中可以包含五元环等。

定义 K_i 为 i 个点的完全图，根据拉姆齐定理，如果黑白两色的子图均不包含三元环，那么黑白子图的并集不能包含 K_6。证明很简单，在完全图中，如果一个点有三条颜色相同的邻边，分别指向 x_1, x_2, x_3，那么由这三个点构成的 K_3 必须都是另一个颜色，不符合题意，所以每个点同一种颜色的邻边最多两条，而在 K_6 中每个点有五条邻边，无法满足这个条件。

因此，黑白子图并集不包含 K_6 是答案的必要条件，在这个必要条件下最多可以选择多少条边，图兰定理给出了一种构造方式：如果想让一张图不包含 K_i 的子图 $2 \leq i$，我们把所有点尽可能平分成 $i-1$ 个集合，每个集合内的点之间不进行连边，属于不同集合的两个点之间均可进行连边，这样的图边数最多。在本题中，我们将点尽可能平均地划入五个点集 S_0, S_1, S_2, S_3, S_4，我们将 S_i 和 $S_{(i+1) \bmod 5}$ 之间的边全部进行黑化，S_i 和 $S_{(i+2) \bmod 5}$ 之间的边全部进行白化（形状类似于五行图），此构造的黑白子图均不包含三元环，符合充分条件；为了不包含 K_6，我们最多只能连这么多边，满足必要条件，此构造就是边数最多的正确答案。

在构造完成后，我们会发现黑边子图按照顺序依次选取相邻点集中的点，总能找到一条长度为 n 的环，从中截取一段长度为 k 的链作为基底即可。

图兰定理的一个简化版证明：

我们假设当前构造的不包含 K_i 的子图为 G，其中存在三个点 x, y, z 满足 x, y 和 y, z 之间无连边而 x, z 之间有连边，我们证明一定可以通过调整使得边数增加。设 d_x 为点 x 的度数，若 $d_y < d_x$，我们可以进行一次「替换」操作，删除 y 的所有连边，对于每一个和 x 连边的点增加一条向 y 的连边，显然若原图不含 K_i，那么替换后的图中也不含 K_i，但总边数增加；做完替换操作后 $d_y \geq d_x$，同理 $d_y \geq d_z$，这时我们再进行两次替换操作，将 x, z 两个点的连边替换成 y 的连边，得到新图 G'，且新图的边数满足：$e(G') = e(G) - (d_x + d_z - 1) + 2d_y \geq e(G)$，减一是由于 x, z 两点原本存在一条连边。通过不断进行如此调整，我们得到一个结论：最终答案的补图是由多个完全子图组成的。由于不包含 K_i，显然这样的完全图最多有 $i-1$ 个，易证当每个完全子图点集尽可能平均时补图的边数最少，此时答案的边数最多。

B. 简单可满足性问题

结论：假设当前有一组赋值满足了 p 个子句，有 $m-p$ 个子句未满足，则将当前的 n 个变量赋值全部取反（0 变 1，1 变 0），至少会使之前未能满足的 $m-p$ 个子句得到满足，因此满足的子句数量大于等于 $m-p$。

由于只需要满足 $\lceil \frac{m}{2} \rceil$ 个子句，因此假设当前一组赋值未能满足 $\lceil \frac{m}{2} \rceil$ 个子句，将该组变量赋值全部取反，则新的赋值一定能至少满足 $\lceil \frac{m}{2} \rceil$ 个子句。

因此该题的解法便有很多种。

- 方法一：随机一组赋值，若该赋值不合法，将全部的变量取反后一定为一组合法赋值；
- 方法二：检验全 0 赋值，合法则输出，否则输出全 1 赋值；

- 方法三：一直随机一组赋值，直到遇到一组合法赋值。期望的随机次数小于 2 次；

还有很多乱搞、贪心的做法，由于时限较宽应该也能通过。

Fun fact：出题人本来想出找到最小字典序的版本，但后来该问题被证明为 NP-hard 问题，导致出题人已经造好的题白造了。

C. 箭串

本题有两种方法：

1. 正向考虑：对于一个箭串分解为前一个 `>`，中间的 `-` 部分和最后的 `>>>`，分三部分用线段树维护区间覆盖即可。但需要注意时限较紧，一些实现可能无法通过。
2. 逆向考虑：考虑到区间覆盖只取决于某个位置最后一次被覆盖到的内容，因此考虑将操作序列逆序，用链表维护没有被覆盖的位置。操作时寻找要覆盖的区间中第一个没有被覆盖的位置，可以用 `set` 维护没被覆盖的位置，在 `set` 中二分即可，之后进行覆盖并进行链表和 `set` 删除操作。

时间复杂度：$\mathcal{O}(m\log n)$，空间复杂度：$\mathcal{O}(n)$。

D. 阿罗祖斯坦的桥

对于一个建立桥梁最多的方案，考虑若在第 l 个和第 $r+1$ 个城市之间连接一条桥梁，那么必然要选择一个划分点 m，$l\le m<r$，然后在第 l 个和第 $m+1$ 个城市之间、第 $m+1$ 个和第 $r+1$ 个城市之间连接桥梁，否则会发现总存在位置可以继续建立桥梁，从而桥梁的数量不是最多的。进而，在这两个部分内部可以继续寻找划分点，直到在相邻两个城市之间连接桥梁。因此，可以发现，若有 $n+1$ 个城市，最多建立的桥梁数量是 $2n+1$ 个，并且这一数量与划分点位置的选择无关。

因此，可以设计 DP 状态 $f(l,r)$ 表示在第 l 个和第 $r+1$ 个城市之间的部分建立桥梁的最小总长度。转移方程如下：

$$f(l,r)=\begin{cases} w(l,r), & l=r \\ \min\limits_{l\le m<r}\{f(l,m)+f(m+1,r)\}+w(l,r), & l<r \end{cases}$$

其中，$w(l,r)$ 表示第 l 个和第 $r+1$ 个城市之间连接的桥梁的长度，即 $w(l,r)=d\cdot\arcsin(\frac{\sum_{i=l}^{r}a_i}{d})$，其中的求和可以前缀和优化。最终，$f(1,n)$ 是题目的答案。事实上，这是一个经典的 区间 DP 问题。直接按照转移方程式进行求解的时间复杂度为 $O(n^3)$，无法通过本题。

区间 DP 问题有一个非常经典的 四边形不等式优化，可以适用于本问题，使得时间复杂度降为 $O(n^2)$，可以通过本题。为了证明四边形不等式优化对于本问题的适用性，需要证明以下两个命题：

- 对于任意 $A\le B\le C\le D$，$w(A,C)+w(B,D)\le w(A,D)+w(B,C)$ 均成立。

令 $x=\frac{\sum_{i=A}^{B-1}a_i}{d}$，$y=\frac{\sum_{i=B}^{C}a_i}{d}$，$z=\frac{\sum_{i=C+1}^{D}a_i}{d}$，则上述不等式等价于

$$\arcsin(x+y)+\arcsin(y+z)\le\arcsin(y)+\arcsin(x+y+z)$$

移项得

$$\arcsin(x+y)-\arcsin(y)\le\arcsin(x+y+z)-\arcsin(y+z)$$

设 $g(t)=\arcsin(x+t)-\arcsin(t)$，则上式等价于 $g(y)\le g(y+z)$，其中 $x,y,z\ge 0$，$x+y+z\le 1$。等价于证明 $g(t)$ 单调递增。求导证明即可。

- 对于任意 $A\le B\le C\le D$，$w(B,C)\le w(A,D)$ 均成立。

根据 $\arcsin$ 函数的单调性也是易证的。

E. 猫猫城大选（困难版）

维护两个栈，分别维护当前差值是正的和负的的所有情况的概率，以及 z 是当前差值为 0 的概率。那么转移相当于把所有情况的值加减 1，可以通过维护偏移量 $offset$ 来 $O(1)$ 实现。总复杂度 $O(n)$。

F. 我知道你姓啥！

竖着看，相当于 m 个长为 n 的字符串。我们的目标是使这 m 个字符串互不相同。一次添加操作相当于给每个字符串末尾添上 0 或 1。对于两个相同的字符串，在其末尾一个添 0，一个添 1，就能让它们不同。

用 `std::map` 统计出原来的 m 个字符串中出现次数最多的串，设其出现次数为 c。一次操作能让 $c \leftarrow \lceil \frac{c}{2} \rceil$，暴力模拟即可。

时间复杂度 $O(nm\log m)$。

G. 猜数游戏

令方程 $l + x \equiv a \pmod p$，则求出 x 后 $l + x$ 即为所求。显然 $x \equiv a - l \pmod p$。注意 $l + x$ 可能不在 $[l, r]$ 范围，需判断有无解。

时间复杂度：$\mathcal{O}(1)$，空间复杂度：$\mathcal{O}(1)$。

H. 独立事件

当 A 为空集或全集的时候 A 与所有事件独立，答案即为 2^n。否则 A 与 B 独立的条件可以等价为

$$P(B|A) = P(B|\overline{A})$$

也可以写成

$$\frac{P(AB)}{P(A)} = \frac{P(\overline{A}B)}{P(\overline{A})}$$

可以直观得看成 B 在 A 中与在 $\overline{A}$ 中的占比是一样的。

记 f_i 表示在 A 事件对应集合的子集中选出元素总和为 i 的选法，g_i 表示在 $\overline{A}$ 事件对应集合的子集中选出元素总和为 i 的选法，求出 f_i 与 g_i 的过程是比较入门的 dp。

那么答案是

$$\sum_{i=0}^{1000}\sum_{j=0}^{1000} f_i \cdot g_j \cdot [\frac{i}{P(A)} = \frac{j}{P(\overline{A})}]$$

I. 圆与直线交点

直接用浮点理论上应该会被卡精度。一种做法是使用 Python 的分数类 `Fraction`。使用之后就是纯模拟求出圆心，然后求解。

考虑四个点 $a(x_1, y_1), b(x_2, y_2), c(x_3, y_3), d(x_4, y_4)$，四点共圆的条件可以转化为：

$$\begin{vmatrix} x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 & x_2 & y_2 & 1 \\ x_3^2 + y_3^2 & x_3 & y_3 & 1 \\ x_4^2 + y_4^2 & x_4 & y_4 & 1 \end{vmatrix} = 0$$

我们令 $d=p+vt$，然后只要行列式展开之后会得到一个二次方程 $At^2+Bt+C=0$ 。令 $f(t)=At^2+Bt+C$，为方便得到系数 A,B,C 我们可以先通过平移将点 a 移动到原点，然后计算 $f(-1),f(0),f(1)$ 的值。接着我们只需要通过判别式即可判断圆与直线的交点个数，进而得知圆与直线的位置关系。

上述过程可以完全使用整数来处理，但运算过程会爆 `int128` 建议使用 Python。

J. 创建用户

按题意模拟即可。

时间复杂度：$\mathcal{O}(n)$，空间复杂度：$\mathcal{O}(n)$。

K. 哲学之门

虽然哲学子图中大部分点和边位于环上，但是在原图中枚举每一个六元环是比较复杂的，且容易重复计数，我们选择一个不重不漏的计数方式。

先枚举环外伸出那一条边，这条边连接的两个点度数都必须是 3，易证包含这条边的哲学子图不超过两个。我们分别选择两个点作为环上的起始点，然后给已经固定的六个点（选择的边连接的两个点以及这两个点的邻点）打上 vis 标记，并开始枚举环上剩下的三个点，由于每个点度数不超过三，这个枚举的复杂度相当低，但要注意每枚举一个点都需要判断这个点是否被打过 vis 标记，以及枚举后立即打上标记防止某个点被枚举两次。

此题正解代码很短，主要的易错点就在于 vis 标记上，因为我们枚举的九个点必须保证两两不重复。

L. 乒乓

容易发现度数为 1 的点只有两种情况：和另一个度数为 1 的点相连；和一个大连通块连边。

于是可以对这两种情况进行分类讨论，枚举第一类点的个数 $2i$：对于第一类，分成 i 组，组内连边；第二类点直接向剩下的点连。然后再把没用过的边分配给度数不为 1 的点构成的图。

令 $f(n,m,x)$ 表示**钦定** x 个点度数为 1 且带有重复方案的方案数，$g(n,m,k)$ 表示恰好有 k 个点度数为 1 的方案，那么

$$
\begin{aligned}
f(n,m,x) &= \binom{n}{x}\sum_{i=0}^{\lfloor\frac{x}{2}\rfloor}\binom{x}{2i}\frac{(2i)!}{i!2^i}(n-x)^{x-2i}\binom{\binom{n-x}{2}}{m-x+i} \\
&= \sum_{k=x}^{n}\binom{k}{x}g(n,m,k)
\end{aligned}
$$

反演即得答案

$$
g(n,m,k)=\sum_{x=k}^{n}(-1)^{x-k}\binom{x}{k}f(n,m,x)
$$

复杂度 $O(n^2)$。

M. 前兜车一转后兜车一转

可以先计算出最初序列的价值，这样只用考虑翻转改变的价值。

对于单次翻转，发现只会在区间端点处改变价值，因此可以 $O(1)$ 计算出单次翻转后改变的价值，令前缀为 i。具体地，$\Delta=|a_1-a_{i+1}|-|a_i-a_{i+1}|$ ，后缀同理。

对于两次翻转，我们令前缀为 i，后缀为 j。

当 $i+1<j$ 时，两次翻转对于价值的改变互不影响，对于每个 i 找到价值最大的 j 即可，通过反向枚举可以做到 $O(n)$。具体地，$\Delta=|a_1-a_{i+1}|-|a_i-a_{i+1}|+|a_n-a_{j-1}|-|a_j-a_{j-1}|$。

当 $i+1=j$ 时，这种情况只有 $O(n)$ 种，全部枚举出来即可。具体地，
$\Delta=|a_1-a_n|-|a_i-a_{i+1}|$ 。

当 $i+1>j$ 时，可以找到 j 对应翻转后的下标，正向枚举可以做到 $O(n)$ 。具体地，
$\Delta=|a_1-a_{i+1}|-|a_i-a_{i+1}|+|a_n-a_{i+1-j+1}|-|a_{i+1-j}-a_{i+1-j+1}|$ 。

当 $i,j\in\{1,n\}$ 时直接代入式子可能会越界，拿出来单独计算即可。

单组数据复杂度 $O(n)$，注意多测清空以及答案的值域会超过 `int`。

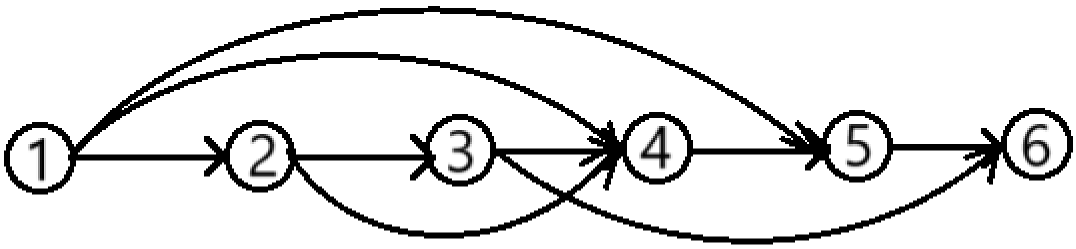
N. 幸运之环 II

首先我们需要找出基环树中的环，可以采用类似拓扑排序的方式在树上找到此环，也可以利用并查集等方法找到使得树成环的边，进而找到此环。之后找到环上编号最小的一点，之后判断顺逆时针方向，取下一个编号更小的方向输出即可。

时间复杂度：$\mathcal{O}(n)$，空间复杂度：$\mathcal{O}(n)$。

O. 不走回头路

结论：将有向图 G 的强连通分量（即所有点都可以互相可达的极大点集）缩点后，形成的图 $G\prime$ 为一个有向无环图。若 $G\prime$ 的一个生成子图为从某一个点出发，连接所有点的一条链，则该图有解，否则无解。



如上图所示，存在 $1\to2\to3\to4\to5\to6$ 的链接所有点的单向链，此为有解情况。

缩点部分可以用 Tarjan 算法实现，判断是否存在连接所有点的单向链可以在缩点后的 DAG 上 dp 实现。

时间复杂度为 $O(n+m)$。

P. 相同字符

预处理 $pre[i]$ 代表从前向后在 a 中匹配上 b 中的字符的个数，即 $a[0..i]$ 能匹配上 $b[0..(pre[i]-1)]$
。

预处理 $suf[i]$ 代表从后向前在 a 中匹配上 b 中的字符的个数，即 $a[i..n-1]$ 能匹配上
$b[(m-suf[i])..m-1]$。

上述的匹配不区分大小写。

若 a 中无大写字母，判断 $\max(suf[0],pre[n-1])=m$ 是否成立即可。

否则枚举 b 中每一个与大写字母相同的位置 i，判断 $pre[i-1]\geq i$ 和 $suf[i+1]\geq m-i-1$ 是否成立。

时间复杂度 $O(n)$。

Q. 校车

容易发现除了车站之外的点是无用的，先使用 Floyd 算法求得两两之间的最短路，这一部分 $O(n^3)$。

之后令 $dp(u,S)$ 表示当前在第 u 个车站，且访问过的车站集合**至少**是 S 时的最短路。对于转移，S 显然不会变小，对于 S 相同的状态，也不会互相转移，因为这是不优的。所以转移不会有环。枚举下一步到达的点，更新状态，由于并不知道具体走过了哪些点，所以只是保证至少走过了 S 中的点。这一部分复杂度 $O(w^2 2^w)$。

之后令 $D(S)$ 表示至少走过 S 中的点的最短闭合路径，可以通过 $dp(u,S)$ 得到。这一部分 $O(w2^w)$。

那最后只考虑如何合并路径即可，相当于一个背包合并。通过枚举子集的技巧，可以做到 $O(K3^w)$。类似二进制分解的思想来合并可以做到 $O(\log K \cdot 3^w)$。

本题时限很宽，std 在评测机上只用 100ms。可能有 A* 高手也能通过此题。