第 50 届 ICPC 国际大学生程序设计竞赛
亚洲区域赛（西安）题解

2025 年 10 月 19 日

在 ICPC 现场赛中，各题通过情况如下表所示（共 417 支队伍有提交，含打星队伍）：

G	L	J	F	I	B	M	C	K	A	D	E	H
99.8%	97.6%	78.2%	59.2%	40.5%	15.3%	12.5%	17	12	8	0	0	0

A. Azalea Garden

先考虑不修改怎么做。

首先找出攻击力最大的生物（如果有多个，任取一个）。设这个生物能击败的生物数目（除了自己）为 x，则答案一定为 x 或者 $x+1$，此时我们只需要判断这个攻击力最大的生物能不能在 x 个生物均被击败的情况下被击败即可，设攻击力最大的生物为第 i 个：

- 如果 $a_i < b_i$ 则该生物不能被击败；
- 否则，找到所有无法被击败（即 $b_j > a_i$）的生物 j 中 a_j 最大的那个，如果 $a_j \geq b_i$，则第 i 个怪物可以在击败 x 个生物之后被 j 击败；
- 否则，若要让 i 被击败，那么必须调整 x 个生物被击败的顺序：在这 x 个生物中选择一列生物 $k_1, k_2, \cdots, k_m$，使得 k_j 能击败 k_{j+1}，k_m 能击败 i，并且存在一个无法被 i 击败的生物能够击败 k_1，那么答案才为 $x+1$。

前两种情况容易判断，只需要考虑最后一种情况。显然令击败 k_1 的生物为不能被 i 击败的生物中攻击力最大的那个是最优的，设其攻击力为 A。则容易证明：存在这样的生物序列，当且仅当所有 x 个生物对应的区间 $(b_j, a_j]$ 能够覆盖整个区间 $(A, b_i]$。

于是不修改的情形已经解决，接下来考虑如何处理修改。

首先，注意到 i 不能击败的生物 j 均满足 $b_j > a_i$，而 $b_i \leq a_i$（否则直接按照第一种情况处理），那么这些生物对应的区间 $(b_j, a_j]$ 不会影响到能否覆盖整个区间的判定，也就是说无需只维护 i 能击败的 x 个生物，可以在线段树上维护所有生物的区间 $(b_j, a_j]$。

于是剩下需要维护的内容只有不能被 i 击败的生物的数量以及其中最大的攻击力。只需要预先将所有防御力离散化，用线段树维护每个后缀的生物数量以及最大攻击力即可。

总时间复杂度为 $\mathcal{O}((n+q)\log(n+q))$。

B. Beautiful Dangos

容易判断一开始就漂亮的情况。接下来发现选择重排的区间有限制：左端点左侧没有相邻的团子颜色相同，右端点右侧也没有相邻的团子颜色相同。下文中称这个限制为基础限制。

考虑如何快速判断选择一个符合基础限制的区间进行重排是否可行。粗略的观察是：如果一种颜色超过了大约一半，那么会导致无论如何重排都无法使得这种颜色的团子不相邻，否则存在重排方案。但是还需要考虑区间两端相邻的团子的颜色也会对重排产生限制，具体地，给出如下结论：

定理. 对于一个符合基础限制的区间，定义一种颜色 c 的权值为下面两部分贡献的总和：

- 区间内颜色为 c 的团子个数减去颜色不为 c 的团子个数（即，每个颜色为 c 的团子每个贡献为 1，每个颜色不为 c 的团子贡献为 -1）；
- 对于区间两端点，每个端点外侧如果有团子且团子颜色为 c，那么贡献 1，否则不产生贡献。

那么对该区间重排可行，当且仅当对于每种颜色，这种颜色的权值不超过 1。

证明. 使用归纳法证明：

- 对于长度为 1 的区间，显然成立；
- 对于长度至少为 2 的区间，考虑确定区间内第一个团子的颜色：令第一个团子的颜色为当前三种颜色中权值最大的那个。剩余部分仍然存在解，因为：
  - ◇ 那么另外两种颜色的权值可能不变或者减少 1（只可能减少第二部分贡献），仍然符合限制；
  - ◇ 而选择的这种颜色，因为区间左边界外侧有一个该颜色，所以第二部分贡献增加 1，但是因为少了一种该颜色，所以第一部分贡献减少 1，仍然符合限制。

  按照上述构造方案，总是存在一种合法的重排方案。

□

可以预处理三种颜色数量的前缀和，这样就可以在 $\mathcal{O}(1)$ 的时间复杂度内判断一个区间是否符合条件。当然，如果上述贡献计算方式较难理解，也可以通过分类讨论两端点外侧一个团子的颜色，分别写出对区间内团子颜色出现次数的限制。

如果已经找出最优的选择区间的方案，那么容易构造出一种合法的重排方案：

- 按照上面给出的证明过程，可以直接线性构造出重排方案；

- 已经可以 $\mathcal{O}(1)$ 时间复杂度内根据两端点外侧团子颜色以及区间内每种团子颜色的数量判断是否可行，那么可以从左到右扫描整个区间，每个位置都枚举放哪种颜色的团子，并 $\mathcal{O}(1)$ 判断是否合法，也可以在线性时间复杂度内构造出重排方案。

于是问题转化为找出最短的合法区间。

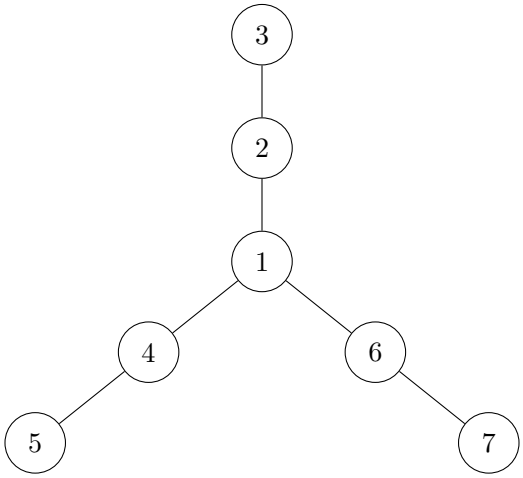
一种简单的处理方法是，二分选择的区间长度，枚举所有这个长度的符合基础限制的区间依次判定。时间复杂度为 $\mathcal{O}(n \log n)$，可以通过此题。

更进一步地，最小的符合条件的右端点随着左端点递增而递增，对于每个左端点双指针求出最小的符合条件的右端点即可做到总时间复杂度 $\mathcal{O}(n)$。

C. Catch the Monster

下称题面中的 *nice* 为*合法*的。

首先考虑合法的图应该满足什么性质。最小的不合法图如下图所示：



注意到一个图只要有一个子图不是合法的，它就不合法。可以用归纳法证明，不存在这个子图的图一定是毛毛虫图，即存在一条简单链，使得每个点到这条链上的距离都不超过 1。

对于毛毛虫图，容易构造一种方案。具体地，按照顺序遍历链上的点，到一个新的链上的点 u 时，对于每个相邻且不在链上的点 v，先选择一次 v，再选择一次 u。容易证明，这样构造的方案符合题目要求。问题转化为如何判断图中是否有这个子结构。

解法一

给出结论：极小的满足存在这个子结构的区间只有 $\mathcal{O}(n)$ 个。考虑枚举中心结点 x，对这个子结构分类讨论，分为 3 条向子树内和 2 条向子树内的。

x 的每个儿子 u 会贡献最多两个区间 $[l_1, r_1], [l_2, r_2]$，即取 u 的儿子中 u 的前驱后继，记他们的颜色为 u。接下来相当于要求极小的区间使得包含至少 k 种颜色，这个可以双指针。对于 2 条向子树内的边构成的子区间 $[l, r]$，只需要取 x 邻域内 l, r 的前驱后继即可。

那么每个点 x 贡献 $\mathcal{O}(|\{u|u \text{ is a son of } x\}|)$ 个区间，加起来就是 $\mathcal{O}(n)$ 个，由于需要离散化儿子，所以这一步时间复杂度 $\mathcal{O}(n \log n)$。最后双指针找出 P_l 表示以 l 为左端点，最大的合法区间右端点。询问可以 $\mathcal{O}(1)$ 回答，这一步时间复杂度 $\mathcal{O}(n+q)$。

总时间复杂度 $\mathcal{O}(n \log n + q)$。

解法二

使用双指针对于每个 l，求出 p_l 表示以 l 为左端点时，最大的使得图合法的 r 的值。问题转化为不断向图中加入一个点或者删除一个点，动态判断图当前是否合法。注意修改操作具备额外性质：每个点只会被插入或删除一次。

前文推理得到，图合法等价于对于每个结点，与其相邻的度数至少为 2 的结点数量小于 3。动态维护每个结点 u 的度数 d_u、与其相邻的度数至少为 2 的结点数量 f_u，以及整张图中违反限制 $f_u < 3$ 的结点数量。

考虑在关于操作结点度数线性的时间复杂度内处理操作。对于加入点的操作：

- 对于所有点的结点度数以及 f_u，只需要遍历所有相邻结点更新即可；
- 对于相邻结点 v 的 f_v 值，只有在 u 度数至少为 2 时，需要更新所有相邻结点的 f 值；
- 对于违反限制 $f_u < 3$ 的结点数量，只需要找到所有可能被 u 影响的结点即可：
  - ◇ 若 u 作为中心结点或者度数至少为 2 的点，那么可以直接枚举中心结点并更新；
  - ◇ 若 u 作为子图中的叶子结点，那么总数只会在其相邻结点度数正好变为 2 时发生变化，这样可能的中心结点的数量至多只有相邻结点个数个，只需要对每个结点额外维护所有相邻结点编号异或和，即可快速求得受影响的中心结点的编号。

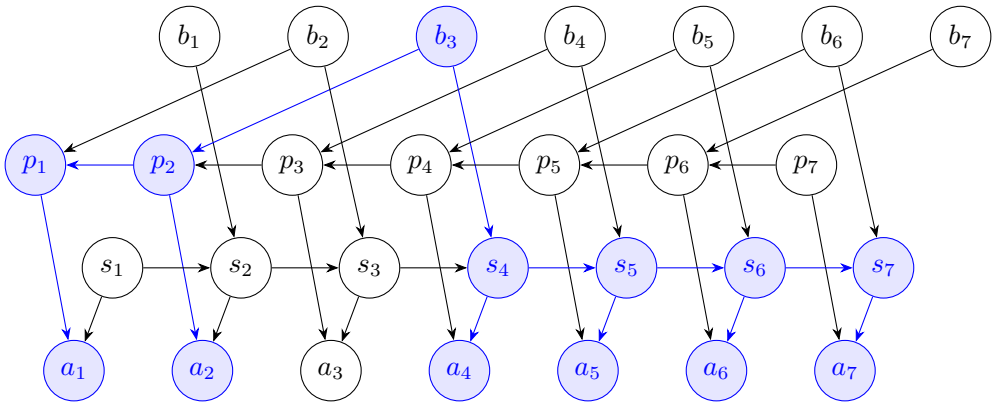
删除点的操作是类似的，此处不再赘述。总时间复杂度为 $\mathcal{O}(n+q)$。

D. Directed Acyclic Graph

观察题目所给的性质：构造的图中点数较大，但是 $m-n$ 较小，并且结点 1 能够到达所有结点。那么找出 DAG 中以 1 为根的外向生成树后，额外的边数量只有 $m-n+1=29$ 条（仅考虑第二个测试点）。不考虑单个点的可达点集的贡献时（这部分至多 n 个，远小于题目要求的 1.6×10^4），题目对于 $m-n$ 的限制显著强于单独对 n 的限制，所以只需要考虑在额外边数固定时，如何构造才能使不同可达点集合数量尽可能多。

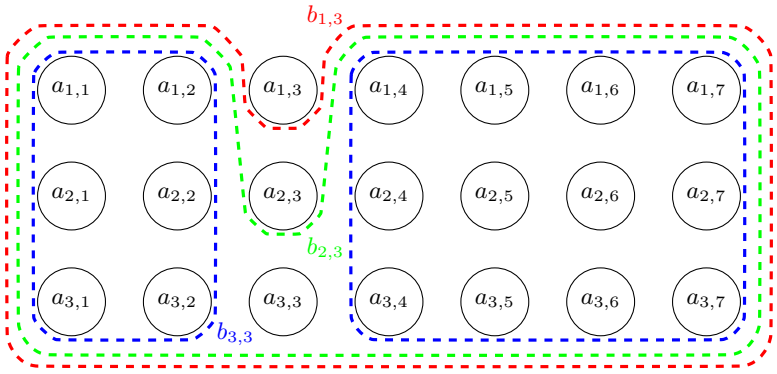
首先给出一种较为容易的构造：取一个正整数 w，对于一行结点 $a_1, a_2, \cdots, a_w$，构造另一行结点 $b_1, b_2, \cdots, b_w$，使得 b_i 能够到达所有满足 $j \neq i$ 的 a_j。这样可以构造出至少 2^w 个不同的可达点集，因为对于每个 a_i，都可以用 b_i 是否在所选的集合中来自由控制 a_i 是否在可达点集中。

朴素建出上面的图较劣。可以令额外的两行结点 $p_1, p_2, \cdots, p_w$ 和 $s_1, s_2, \cdots, s_w$ 分别可达 a 的一个前缀和一个后缀，从而优化所需要的额外边的数量。若取 $w=7$，上述构造方案如下图所示：



例如 b_3 可达的点集如图中蓝色结点所示，包含 $a_1, a_2, \cdots, a_w$ 中除了 a_3 之外的所有结点。该构造方案所需要的额外边数大约为 $3w$（注意还需要计算与 1 相连的边），即可以构造的集合数量为 $\mathcal{O}(n + 2^{\frac{m-n}{3}})$，还不足以通过此题。

考虑拓展上面的构造方案：记两个 d 行 w 列的点阵 $a_{i,j}$ 和 $b_{i,j}$，其中每个 $b_{i,j}$ 都可达 a 中所有点去掉某一列的一个前缀。例如，当 $w = 7$，$d = 3$ 时，$b_{i,3}$ 的可达点集如下图所示：



这样可以构造出至少 $(d+1)^w$ 个不同的可达点集，因为对于每一列，都可以自由选择可达这列的一个后缀。类似地，可以令两行结点 $p_1, p_2, \cdots, p_w$ 和 $s_1, s_2, \cdots, s_w$ 分别可达前缀若干列和后缀若干列即可优化额外边的数量。

上述方案构造的图中需要 $(d+2)w + \mathcal{O}(1)$ 条额外边，取 $d = 3$ 可以得到一个不同可达集合数量为 $\mathcal{O}(n + 4^{\frac{m-n}{5}}) = \mathcal{O}(n + 2^{\frac{m-n}{2.5}})$ 的构造方案，并且可以证明这取到了该方案的最优解，使用其他的 d 值或者每一列混合使用不同的点数量不会更优秀。

具体对于本题的数据范围，取 $d = 3$，$w = 7$ 时最优，不同的可达集合数量至少为 $4^7 = 16\,384$。需要注意两行表示前缀和后缀的点中，边上的 1 个点可以优化掉，上述构造足以通过此题。

E. Epilogue of Happiness

认为题面中的 m 个开关 $o_1, o_2, \cdots, o_m$ 是一个点到根的路径反转，q 张照片是一个点到根的路径查询反转次数为奇数的位置的权值和。

对树进行轻重链剖分（将 o_i 和 x_i 出现次数也算进点权），每条重链上需要处理原问题的序列版本（点到根路径变为序列上的前缀），序列长度是重链长度，修改和查询包含以重链顶为根的子树内的所有修改和查询。

对于序列上的问题，先考虑一个暴力做法：从后往前扫序列，对每个询问记录 v 表示扫到当前位置时，o 在 $[l,r]$ 内且覆盖当前位置的修改个数的奇偶性，s 表示已扫过的位置对答案的贡献（对询问有贡献的前缀内，每扫过一个位置时 $s \leftarrow s+v$），遇到修改时把 $[l,r]$ 包含这个修改的 o 的询问的 v 反转。

可以序列分块，使得块内的点数、修改数、询问数和块数都不超过 B。每一部分贡献分开计算：

- 块内修改对 x 在块内的查询的贡献沿用暴力做法（每块只有 $\mathcal{O}(B^2)$ 对修改查询之间有贡献）；
- 块内修改对 x 在块前面的查询的影响只有整体的反转，可以使用树状树组维护时间轴快速计算反转次数奇偶性；
- 块内修改对 x 在块后面的查询的贡献只有 $\mathcal{O}(B^2)$ 种本质不同的情况（l,r 各有 $\mathcal{O}(B)$ 种情况），可以 $T(B)=2T(\frac{B}{2})+\mathcal{O}(B^2)=\mathcal{O}(B^2)$ 对块内进行序列分治预处理出每种情况的贡献，然后对每个查询可以 $\mathcal{O}(1)$ 计算贡献。
  如果某一个点上修改数和询问数较大，那么无法按照上述方案处理贡献，需要特判。此时块内一定只有一个结点，可以预处理前缀和，对每个询问直接 $\mathcal{O}(1)$ 查询包含的修改个数的奇偶性。

于是可以在 $\mathcal{O}(n^{1.5})$ 的时间复杂度内处理序列上的问题（不妨设 $n=m=q$）。在树链剖分上的总时间复杂度是 $T(n)=\sum\limits_i T(n_i)+\mathcal{O}(n^{1.5})=\mathcal{O}(n^{1.5})$，其中 $\sum\limits_i n_i \le n$ 且 $n_i \le \frac{n}{2}$。

F. Follow the Penguins

对于第 i 只企鹅，称它的目标企鹅为第 t_i 只企鹅。注意到在企鹅 i 停止运动前速度不会改变，而企鹅 t_i 的速度最多变化一次。

维护所有企鹅与目标企鹅相遇的所有事件。每次找出时间最早的那个，这只企鹅 x 记为到达目标且停止运动，此时受影响的事件即为所有目标企鹅为 x 的事件，只需更新这部分事件即可。

总更新次数为 $\mathcal{O}(n)$，可以用堆维护所有事件，时间复杂度为 $\mathcal{O}(n\log n)$。

更进一步地，注意到第 i 只企鹅的停止运动时间只受到第 t_i 只企鹅的影响。于是可以根据影响关系建图，由于一个点至多只有 1 的出度，所以得到的图一定是一个内向基环树森林。对于每一个环，只需要找出按照初始速度计算，最早停止运动的那一个企鹅，即可按照环上的顺序依次计算出所有企鹅的停止运动时间。

时间复杂度为 $\mathcal{O}(n)$，实现时也可以不显式地建出图。

G. Grand Voting

贪心地想，a_i 值较小的人更容易使 s 增加。因此若想要让最终的 s 最大，需要将 a_i 从小到大排序。若想要让最终的 s 最小，则需要将 a_i 从大到小排序。

严格的证明只需要讨论邻项交换后对 s 的影响。根据实现方式不同，时间复杂度为 $\mathcal{O}(n \log n)$ 或者 $\mathcal{O}(n)$。

H. Heart of Darkness

首先考虑算出不限制异色节点之间连边的方案数，再减去异色边小于 k 的树的个数。先考虑计算异色边数恰好为 k 的情况，枚举黑点的数量 i，可以得到其为

$$\sum_{i=1}^{n} \binom{n}{i} i^{i-2} (n-i)! [x^k y^{n-i}] (\mathrm{e}^{xT(y)})^i$$

其中形式元 x 计量异色边数，y 计量白点数，$T(y)$ 是有标号有根树数量的 EGF。那么异色边小于 k 的树的个数就是

$$S_k = n! [y^n] \sum_{j=0}^{k-1} \frac{T(y)^j}{j!} \sum_{i=1}^{\infty} \frac{i^{i-2+j}}{i!} y^i$$

由此也容易得到 $k \to \infty$ 时 S_k 为

$$n! [y^n] \sum_{i=1}^{\infty} \frac{i^{i-2}}{i!} (y\mathrm{e}^{T(y)})^i = n! [y^n] \sum_{i=1}^{\infty} \frac{i^{i-2}}{i!} T(y)^i$$

然后使用另类 Lagrange 反演处理 S_k：

$$S_k = n! [x^n] \mathrm{e}^{nx} (1-x) \sum_{j=0}^{k-1} \frac{x^j}{j!} \sum_{i=1}^{\infty} \frac{i^{i-2+j}}{i!} (x\mathrm{e}^{-x})^i$$

设 a_m 为红色部分的 $[x^m]$ 项系数，则对于 $m \le k$ 时，a_m 显然是 $m^{m-2}/m!$，下面就只讨论 $m > k$ 的情况：

$$\begin{aligned} a_m &= \sum_{j=0}^{k-1} \frac{1}{j!} \sum_{i=1}^{m-j} \frac{i^{i-2+j}}{i!} [x^{m-j}] (x\mathrm{e}^{-x})^i \\ &= \sum_{j=0}^{k-1} \frac{1}{j!} \sum_{i=1}^{m-j} \frac{i^{m-2}}{i!} \times \frac{(-1)^{m-j-i}}{(m-j-i)!} \\ &= \sum_{j=0}^{k-1} \frac{1}{j!} \left\{ {m-2 \atop m-j} \right\} \end{aligned}$$

根据 Stirling 多项式的性质（P8561）可知，a_m 是关于 m 不超过 $2(k-1)$ 次的多项式，也就是说其 GF 是一个分母为 $(1-x)^{2k-1}$ 的有理分式。所以在计算出 a_k 的前 $(2k-1)$ 项之后，就可以用一次卷积快速求出其 GF。

现在就有：

$$S_k = n![x^n]\frac{\mathrm{e}^{nx}}{(1-x)^{2k-1}}P(x)$$

其中 $P(x)$ 是一个 $\Theta(k)$ 次多项式（可能比分母高 2 次以内），此时可以直接以 $\Theta(n)$ 的时间复杂度计算 S_k。

总时间复杂度 $\Theta(n+k^2)$。做到这个复杂度需要线性筛出 $f(i)=i^i$ 的函数值，但时间常数较大，朴素地使用快速幂也是可以通过的。

I. Imagined Holly

首先观察到对于三个结点 x,y,z，可以通过 $a_{x,y}\oplus a_{y,z}\oplus a_{z,x}$ 找到三者的唯一中心点的编号，使得以该中心点为根时，x,y,z 在三个不同的子树中。

取结点 1 作为树的根。此时，对于任意两个结点 x,y，x 是 y 的祖先当且仅当 $1,x,y$ 的中心点为 x。这一点容易证明。

于是我们可以在 $\mathcal{O}(n^2)$ 的时间复杂度内找到所有的祖先关系，进一步地，可以求得每个结点的深度。而一条边存在，当且仅当两个点有祖先关系且深度正好相差 1，再次遍历所有祖先关系，即可根据深度找出所有树边。

根据上面的算法流程可知，符合题意的 *holly tree* 是唯一的。总时间复杂度为 $\mathcal{O}(n^2)$。

J. January's Color

由于不能主动丢弃结点，所以只能通过与其它结点合并，生成父结点的方式来丢弃结点。

这意味着当初始只拥有结点 x 时，只能通过与兄弟结点合并来变为只拥有父结点的状态，通过若干次这样的操作，最终变为只拥有结点 y。所以当 x 不在 y 的子树中时无解；否则，最小花费为从 y 到 x 的路径上，每个点转化为其父亲点的最小花费之和。下面只需要考虑对于每个结点，将其转化为其父亲结点的最小花费。

要将一个结点转化为其父亲结点，就要获得一个兄弟结点与其进行合并操作。而兄弟结点的获得方式没有限制，可以直接购买，也可以通过合并。于是我们可以先进行一次树形 DP，计算出 f_u 表示初始时不拥有任何 u 子树中的结点时，获得结点 u 的最小花费是多少。转移只需要考虑对于每个结点，计算出其儿子结点中 f 值最小以及次小的儿子结点即可。

那么将一个结点转化为其父亲结点，只要找到其兄弟结点中，f 值最小的那个，可以通过之前求出的父亲结点的最小以及次小的儿子结点快速得到。而计算一条链上所有转化的最小花费和，只需要预处理每个结点 u 转化到根结点的最小花费 g_u，就可以通过 g_x-g_y 快速求出答案。

总时间复杂度为 $\mathcal{O}(n+q)$。

K. Killing Bits

首先特判 a 与 b 完全相等的情况。接下来，如果想要让 a 变成 b，至少需要进行一次操作。容易发现有下面的必要条件：

- 对于所有 $1 \le i \le n$，$a_i \,\&\, b_i = b_i$；
- 存在一个排列 p，使得对于所有 $1 \le i \le n$，$p_i \,\&\, b_i = b_i$。

容易通过构造得出这个条件同时是充分的。第一个条件可以直接判断，第二个条件可以用网络流解决：

- 若 u 在二进制下为 1 的位的集合严格包含 v 的，那么 u 向 v 连一条边；
- 源点向 $0, 1, \ldots, n-1$ 各连一条边；
- 汇点从 $b_1, b_2, \ldots, b_n$ 各连一条边。

所有容量均为 1，只需要判断是否满流。如果实现较好可以直接通过。

更进一步地，可以考虑优化建图，每一个二进制数均向自己少掉任意一个二进制位的数连一条容量为 $+\infty$ 的边，仍然判断是否满流。该建图方案的总边数为 $\mathcal{O}(n \log n)$，可以通过此题。

L. Let's Make a Convex!

首先有结论：能把选出的木棍拼接成凸多边形，当且仅当最长的木棍的长度小于其余木棍长度的总和。

将输入的数排序。下文中认为 $a_1 \le a_2 \le \cdots \le a_n$。

对于木棍数量 k，我们希望找到一个最大的 p_k（$k \le p_k \le n$），使得 $a_{p_k-k+1}, a_{p_k-k}, \cdots, a_{p_k}$ 能拼接成凸多边形，若不存在，记 $p_k = 0$。

注意到木棍数量较少的 p_i 一定不会超过木棍数量较大的非零的 p_j。于是可以倒序求解 $p_n, p_{n-1}, \cdots, p_3$，用一个指针维护当前可能的答案中最大的编号。

时间复杂度为 $\mathcal{O}(n \log n)$，瓶颈在于排序。

M. Mystique as Iris

首先考虑判断一个确定长度为 n 的序列 a 是否是神秘的。

注意到如果 $a_1 = 1$，可以归纳证明：当且仅当 a 中所有元素均为 1 且 n 为奇数时，a 不是神秘的。这启发我们去构造一个处在边界上的 1。

如果序列中存在某个元素的值在区间 $[2, n-1]$ 中，那么我们总能通过一系列操作，将这个数移动到边界并转化为 1。此时若序列全为 1 且 n 为奇数，则说明边界上本就有 1，但序列并非全为 1，因此该序列依旧是神秘的。于是可以得出结论：只要序列中存在值在 $[2, n-1]$ 中的数，序列必然是神秘的。

否则，序列 a 仅由若干个 1 和若干个 $+\infty$ 构成。如果 $a_1 = 1$ 或 $a_n = 1$，那么显然 a 是神秘的；如果存在某个 i 使得 $a_i = 1$ 且 $a_{i+1} = 1$，那么可以删除前缀 $a[1, i]$，剩余部分就能转化成边界含 1 的情形，因此 a 依然是神秘的；否则无论如何操作，序列始终保持 $a_1 = +\infty, a_n = +\infty$ 且不存在相邻两个 1 的形式，此时 a 不是神秘的。需要特别注意的一个边界情况是：**当 a 中所有元素均为 1 且 n 为奇数时，a 不是神秘的**。

然后考虑计数 a 可以转化为多少个神秘序列。利用容斥的思路，可以转而统计不神秘的序列。此时序列中的每个元素只能是 1 或 $+\infty$。因此如果原序列中存在 $[2, n-1]$ 的数，答案为 0。而对一个 -1 位置，填 1 有 1 种方案，填 $+\infty$ 有 $\max(m-n+1, 0)$ 种方案。

设 $f_{i,0/1}$ 表示填完前 i 个数，且最后一个数为 $1/+\infty$ 的方案数。状态转移可以直接写出，时间复杂度为 $\mathcal{O}(n)$。最后需要注意，如果序列有可能全为 1 且 n 为奇数，那么答案还需要额外加 1。