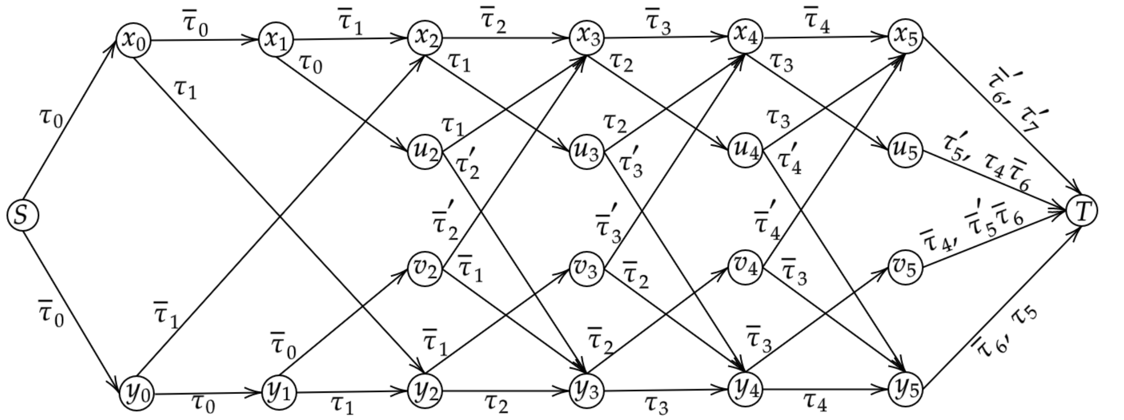
Problem A. Anti-hash Test

考虑建出第m个 *Thue-Morse words* 的后缀自动机，把所有只有一个出度的点都缩掉之后，可以发现很有规律。但是这个规律在比较小的m $(m \le 5)$ 中不存在。因此对于比较小的m用暴力的方法做，对于其他比较大的m，建出缩点之后的后缀自动机。比如下图是$m = 7$的缩点后的后缀自动机。



具体来说最左边是起始节点S，最右边是终点T。如果把上面一排依次定为$x_0, x_1, \ldots, x_{n-2}$，下面一排依次定为$y_0, y_1, \ldots, y_{n-1}$， 中间分别为$u_2, u_3, \ldots, u_{n-2}$和$v_2, v_3, \ldots, v_{n-2}$。那么这些有向边上对应的字符串是:

$x_i \xrightarrow{\overline{\tau}_i} x_{i+1}, \quad x_i \xrightarrow{\tau_{i-1}} u_{i+1}$

$y_i \xrightarrow{\tau_i} y_{i+1}, \quad y_i \xrightarrow{\overline{\tau}_{i-1}} v_{i+1}$

$u_i \xrightarrow{\tau_{i-1}} x_{i+1}, \quad u_i \xrightarrow{\tau'_i} y_{i+1}$

$v_i \xrightarrow{\overline{\tau}_{i-1}} y_{i+1}, \quad v_i \xrightarrow{\overline{\tau}'_i} x_{i+1}$

其中τ_i是第i个 *Thue-Morse words*，$\overline{\tau}_i$是τ_i每个字符取反后的结果，$\tau'_i = \overline{\tau}_{i-2}\overline{\tau}_{i-1}$，$\overline{\tau}'_i$是$\tau'_i$每个字符取反后的结果。

并且最后从$x_{n-2}, y_{n-2}, u_{n-2}, v_{n-2}$连向$T$的边上字符串有点特殊，具体为:

$x_{n-2} \xrightarrow{\overline{\tau}'_{n-1}} T, \quad x_{n-2} \xrightarrow{\tau'_n} T$

$y_{n-2} \xrightarrow{\overline{\tau}_{n-1}} T, \quad y_{n-2} \xrightarrow{\tau_{n-2}} T$

$u_{n-2} \xrightarrow{\tau'_{n-2}} T, \quad u_{n-2} \xrightarrow{\tau_{n-3}\overline{\tau}_{n-1}} T$

$v_{n-2} \xrightarrow{\overline{\tau}_{n-3}} T, \quad v_{n-2} \xrightarrow{\overline{\tau}'_{n-2}\overline{\tau}_{n-1}} T$

还可以观察到，$T, x_{n-2}, y_{n-3}, x_{n-4}, y_{n-5}, \ldots$这些点是后缀自动机的接受态。

于是我们可以用题目给出的字符串在这个后缀自动机上走。可以发现，仅需要保留$O(\log n)$个节点就可以了。最终走到某个节点之后，我们需要求出它的Right集合大小。注意到Right集合大小等价于从这个节点开始能够走到的接受态的路径条数。根据这个图，我们可以列出递推式，用矩阵乘法或者找规律，很容易求出Right集合的大小。这样就解决了第一问。

对于第二问，则需要求出其他节点的Right集合大小，可以发现Right集合大小随着层数的变大是递减的，我们可以暴力求出前面每一层的Right集合大小，以及这个节点对应了多少个字符串。第二问也顺利解决了。

Problem B. Network Test

1 题意

给定一个无向连通图（可能有重边），用尽可能少的生成树覆盖所有边。

2 解法

我们解决原题的一个等价问题：将边集划分为最少的森林（此时森林的个数称为图的**荫度**(arboricity)）。显然，对于连通图，其答案等于原问题的答案。

解决该问题的算法如下：

1. 令当前森林集合$\mathcal{F} = \{F_1\}$，其中$F_1 = \{e_1\}$；
2. 设当前森林集合为$\mathcal{F} = \{F_1, F_2, \cdots, F_k\}$。令记号$c'$表示$(c \bmod k) + 1$;$F_i(uv)$表示森林$F_i$中$u, v$两点间的唯一路径（若不存在则为空集）。从第二条边开始，我们依次加入每一条边。当加入边e_i时：
   (a) 建立辅助有向图，其中顶点集合为$\{e_1, e_2, \cdots, e_i\}$，$e_u$到$e_v$有边当且仅当$e_v \in F_{c'}(e_u)$，且满足$e_u \in F_c, e_v \in F_{c'}$ 或 $e_u = e_i, e_v \in F_1$。若$e_u \in F_c$满足$F_{c'}(e_u) = \varnothing$，则称$e_u$为终止节点。
   (b) 若上图中存在e_i到任意终止节点的路径，使用BFS求出其中**最短**的路径（称为增广路），并将路径中的每一条边移动到下一个森林中（称为增广）；
   (c) 否则，新建森林$F_{k+1} = \{e_i\}$，并令$\mathcal{F} \leftarrow \mathcal{F} \cup \{F_{k+1}\}$。
3. 当前$\mathcal{F}$即为将边集划分为最少的森林的一种方案。

假设在某一步中，找到的增广路为$\epsilon_0, \epsilon_1, \epsilon_2, \cdots$。要证明上述算法的正确性，我们需要证明以下两点：

1. 上述算法找到的**最短**路径确实是合法的增广路（即沿着路径增广后，所有森林依然是森林）。

   证明： 对于森林F_c， 我们考虑移入该森林的边$\epsilon_u, \epsilon_{u+k}, \cdots, \epsilon_{u+nk}$和移出该森林的边$\epsilon_{u+1}, \epsilon_{u+k+1}, \cdots, \epsilon_{u+nk+1}$（不考虑增广路中的最后一条边）。我们依次将$\epsilon_u, \epsilon_{u+k}, \cdots, \epsilon_{u+nk}$加入$F_c$；在添加$\epsilon_{u+ik}$时，我们将$\epsilon_{u+ik}$两端点间的未标记边（即割边）标记为$i$，则根据BFS的性质，$\epsilon_{u+ik+1}$一定具有标记$i$（注意，此时删去任意具有标记$i$的边，其他具有标记$i$的边将变成割边）。然后，我们依次移除$\epsilon_{u+nk+1}, \cdots, \epsilon_{u+k}, \epsilon_u$，由前述注意可知，所有被标记边都被还原为割边，因此增广后的F_c依然是森林。

2. 若上述算法未找到增广路，则无法将前i条边划分为k个森林。

   证明：令S_i表示BFS结束后，F_i中所有被访问过的边的边导出子图。

   下面我们证明所有S_i的顶点集合都是相同的，即$V(S_1) = V(S_2) = \cdots = V(S_k) = S$。注意到对于$S_i$中的每条边$uv$，在$S'_i$中$u$和$v$之间的路径都会被标记。因此，$V(S_i) \subseteq V(S'_i)$，进而有$V(S_1) \subseteq V(S_2) \subseteq \cdots \subseteq V(S_k) \subseteq V(S_1)$，从而$V(S_1) = V(S_2) = \cdots = V(S_k) = S$。

   然后，我们证明每个S_i都是连通图。由上一段的论证可知，若u, v在S_i中连通，它们在$S_{i'}$中也连通（因为$F_i(uv)$上的每条边在$S_{i'}$中都对应这一条路径）。假设u是边e_i（即新加入的边）的一个顶

点。我们需要证明，$S_1, S_2, \cdots, S_k$中的任意边都和u连通。假设某个S_i中存在某条被访问过边与u不连通，令$\epsilon_t \in S_{i'}$为最早被访问的这样的边。由归纳假设知ϵ_{t-1}在S_i中与u连通，因此ϵ_{t_1}的端点与在$S_{i'}$中与u连通，又因为$\epsilon_t \in F_{i'}(\epsilon_{t-1})$，因此$\epsilon_t$在$S_{i'}$中也与$u$连通，矛盾。

因此，$S_1, S_2, \cdots, S_k$都是大小为$|S|$的树。由于e_i的两端点都在S内，在仅考虑前i条边时，原图的S-点导出子图就有$k(|S|-1)+1$条边，因此不可能将当前图的边集划分为k个森林。

由于最多尝试增广m次，在实现时保证每次尝试增广时每条边至多访问一次，总复杂度即为$O(m^2)$。

Problem C. Mine Sweeper

- 如果 $S \le 24$，我们可以构造这样的地图：“.X.X.X$\cdots$”，可知当长度为 l 的时候，数字和就等于 $l-1$。

- 如果 $S > 24$，我们可以把 S 写成 $S = 8a + 3b$ 的形式，其中 $a, b \ge 0, b < 8$，那么我们可以构造类似如下的地图：

  ....
  .X.X.X.X.
  ....
  .X.X.....
  ....
  ....
 XX

  其中包含恰好 a 个孤立的 “X” 和 b 个右下角那种连续的 “X”，可知一个孤立的 “X” 可以对数字和带来 8 的贡献，而右下角那种连续的 “X” 一个可以带来 3 的贡献。

Problem D. Permutation Counting

我们根据neighbour sequence来构造原排列. 比如对于 0 0 1 1, 我们先放下一个数字 1:

1

然后读 neighbour sequence 的第一个数字, 是 0, 那么我们接着在 1 的右方写下 2:

1, 2

接下来 neighbour sequence 的下一个数字仍然是 0, 那么我们在 2 的右方写下 3:

1, 2, 3

接下来 neighbour sequence 的下一个数字是 1, 那么我们在 3 的左方写下数字 4, 这时候发现一共有 3 种情况 (4 1 2 3), (1 4 2 3), (1 2 4 3), 我们随便挑一个, 比如 (4 1 2 3)

4 1 2 3

接下来 neighbour sequence 的下一个数字是 1, 那么我们在 4 的左方写下数字 5:

5 4 1 2 3

我们得到的这个序列, 假设叫做 c 吧, $c_1 = 5, c_2 = 4, c_3 = 1, c_4 = 2, c_5 = 3$, 我们构造一个排列 a, 使得 $a_{c_i} = i$, 比如这里就是 $a_1 = 3, a_2 = 4, a_3 = 5, a_4 = 2, a_5 = 1$, 可以发现这样构造的序列 a 的 neighbour sequence 恰好等于给出的序列, 而排列 c 和排列 a 是一一对应的. 于是我们只要数满足条件的序列 c 的个数即可.

令 f[i][j] 表示现在已经放到第 i 个数字, 而且第 i 个数字放的位置是 j. 当前状态是 f[i][j] 的话, 如果输入序列的第 i 位是 0, 就需要往 f[i + 1][k] $(k > j)$ 转移, 否则需要往 f[i + 1][k] $(k \le j)$ 转移. 利用前缀和优化一下 DP 即可做到 $O(n^2)$.

Problem E. Treecutting

试图枚举删除点之后的树的直径中心, 可能是一个点也可能是一条边, 两种情况都要枚举一下.

对于点的情况, 假设枚举的点是 u, 那么我们需要删除的所有点就是和 u 距离超过 $\lfloor \frac{k}{2} \rfloor$ 的点数. 我们现在就是要计算出对于树上每个点, 与其距离超过 $\lfloor \frac{k}{2} \rfloor$ 的点数.

使用点分治, 假设现在点分治过程中的根为 r, 我们就是要计算对于树内每个节点, 与其分属在 r 不同子树内, 距离超过一个定值的点数. 这可以简单地计算出每个节点到根 r 的距离, 对于每棵 r 的子树做后缀和统计即可.

总复杂度是 $O(nlogn)$.

边的情况类似.

Problem F. Divide and Conquer

首先可以随便找一条线 l_1 把 n 个点分成两半，不妨令这条线的斜率接近于 0，然后可以在 $[0, \pi]$ 之间二分斜率 θ，可以求出在 θ 斜率下的平分线 l_2，如果同时在 l_1, l_2 这两条线上方的点 $\ge \frac{n}{4}$ 则 $r = \theta$，否则 $l = \theta$，找到临界角度之后取经过相对位置排名 $\frac{n}{2}, \frac{n}{2} + 1$ 的两个点的直线，再微调一下就可以构造出第二条直线了。

Problem G. Coin Game

我们可以发现, 每个 machine 给予的硬币可以等价成, 一个重量为 1 的价值为 a_i 的硬币, 以及一个重量为 2 的价值为 $a_i + b_i$ 的硬币, 这两硬币之间没有相互影响. (对于第 i 个 machine, 如果只取一个, 等价于取重量为 1 的那枚, 如果取了前两个, 等于取了重量为 2 的那枚, 如果三个都取了, 等价于同时取了重量为 1 和重量为 2 的那枚).

现在问题就变成了, 有 n 枚重量为 1 的硬币和 n 枚重量为 2 的硬币, 你现在要求恰好取总重量为 k 的硬币的最大价值.

我们可以按性价比从大到小排序, 然后选性价比最高的那些, 直到取到的硬币总重量不小于 k. 如果这时候取的硬币总重量恰好为 k, 那么这一定就是最优取法. 否则就是总重量恰好为 $k + 1$, 这时候我们需要丢弃一枚性价比最低的重量为 2 的硬币, 拿多一枚重量为 1 的硬币.

根据这种做法, 可以发现, $f(k+1)$ 的最优解一定是 $f(k)$ 的基础上, 多拿一枚重量为 1 的硬币, 或者丢掉一枚重量为 1 的硬币并且多拿一枚重量为 2 的硬币, 这意味着从 $f(k)$ 的最优解转移到 $f(k+1)$ 的最优解是 $O(1)$ 的. 而 $f(1)$ 的最优解一定是拿重量为 1 的价值最大的那枚硬币. 于是整个过程就是 $O(nlogn + m)$

Problem H. I do not know Graph Theory!

如果一开始不强连通，由于竞赛图DAG是一条链，翻转后强连通当且仅当点i在DAG头且点j在DAG尾。

如果一开始强连通，由于强连通竞赛图必有哈密顿回路，翻转后不强连通的边一定在回路上。

设$S_{i,j}$为所有从某个[回路上从i到preivous(j)会经过的点]到某个[回路上从j到preivous(i)会经过的点]的边，可以发现某个边翻转后不强连通当且仅当它是某个$S_{i,j}$的唯一元素。考察$|S_{i,j}|$，发现每条边对其的贡献是一个循环矩形，扫一遍前缀和即可。

Extra: 1.每个$|S_{i,j}| = 1$就是一个scc，其实可以问翻转后scc size的倒数和之类的。 2.尝试构造数据，使得一开始强连通，且翻转后不强连通的边有（至少）p条。

Problem I. Photography

注意到

$$S(x) = \frac{1}{|P|}\sqrt{\sum_{i\in P}\sum_{j\in P}(x_i - x_j)^2} = \frac{\sqrt{2}}{|P|}\sqrt{|P|\sum_{i\in P}x_i^2 - \left(\sum_{i\in P}x_i\right)^2} = \sqrt{2}\,\mathrm{Var}(x)$$

即我们需要最大化投影点的坐标的方差。由主成分分析可知，方差的最大值等于所有点坐标的协方差矩阵

$$\begin{bmatrix} \mathrm{Var}(x) & \mathrm{Cov}(x,y) \\ \mathrm{Cov}(x,y) & \mathrm{Var}(y) \end{bmatrix}$$

的较大的特征值。因此我们可以维护$\sum x, \sum x^2, \sum y, \sum y^2, \sum xy$，这样可以快速计算$x$和$y$的方差以及它们的协方差，从而在$O(1)$时间内求出协方差矩阵的特征值。

Problem J. I do not know Game Theory!

Alice 第一轮取完后，Bob 上来取取同行或者同列必败，因此一定取不同行不同列的。本题性质允许行交换列交换，不妨设 Alice 取 $(1,1)$, Bob 取 $(2,2)$。

这样除 $(3,3)$ 外，剩下的每个格子都是不论后续局面如何，谁第一个取最后一颗石子谁必败。因此 $(3,3)$ 整堆与其他每堆除一颗石子外的整堆构成一个普通 nim 游戏，即后手必败当且仅当 $((1,2)-1)\oplus((1,3)-1)\oplus((2,1)-1)\oplus((2,3)-1)\oplus((3,1)-1)\oplus((3,2)-1)\oplus(3,3)=0$。

Extra:

我不知道怎么任意构造答案不是 7 与 9 的数据。

为什么一上来要拿走整堆？当然是因为我不会一般情形啊（

Problem K. Task Scheduler

目标是最小化

$$\sum_{i=1}^{n}\frac{\binom{t_{p_i}}{m-\sum_{j=1}^{i-1}t_{p_j}}}{\binom{t_{p_i}}{m-k-\sum_{j=1}^{i-1}t_{p_j}}}$$

结论是如果$k>0$，则将工作任务按需要机器数量降序排序，否则答案是$1,2,\ldots,n$，证明留作练习。