Problem Tutorial: "Aqours"

给出的点可以视为是按照 BFS 序给的，也就是说从浅到深给出。可以再给每个节点 u 维护一个 f_u 值，表示离 u 最近的叶子节点到它的距离。

所以每当扫到一个叶子节点，就可以暴力往根节点跳，边跳边更新 f 值，直到跳到一个已被其他叶子节点跳到过的节点为止。

那么对于当前的叶子节点，离它最近的编号小于它的叶子节点到它的距离就是跳到这个终止节点的 f 值 + 跳的步数。

在求完之后，还要从上述的终止节点沿着原路更新一下 f 值，因为可能当前叶子比较深但之前有比较浅的叶子节点。

因为点是按从浅到深的顺序给出，所以一个节点的 f 值只会被最先跳到它的叶子节点赋值一次，也只会被更新一次，所以复杂度是 $O(n)$ 的。

Problem Tutorial: "玖凛两开花"

考虑二分答案是否>= x，可以把图变成一个二分图，左边是< x的，右边是>= x的。在原图中两部分内部是可能有边的，但是选出这些边不会对"答案能达到x"更有利，所以这些边忽略即可。然后可以用匈牙利算法检查是否左边的点是否全部都能匹配上。时间复杂度$O(nmlogn)$，因为匈牙利算法很难卡到上限$O(nm)$且常数很小，所以也能通过。

实际上二分答案的过程可以去掉。我们考虑把二分的过程变成从小到大枚举，到$x+1$的时候就是把x点加入二分图的左边。如果x之前在右边的时候和某个左边的点v匹配过了，把这个匹配关系拆掉，把v再匹配一遍就可以了。时间复杂度$O(nm)$。

如果用Dinic或者HK的话，时间复杂度可以做到$O(m\sqrt{n})$。

Problem Tutorial: "御坂妹妹"

求出两个半平面交，一个是所有直线上方的半平面交，一个是所有直线下方的半平面交，然后再用扫描线从左到右扫一遍即可。

实际上也不需要半平面交的前置知识，把所有直线按斜率排序，用单调栈算出每条直线作为最大值/最小值的横坐标区间即可。

Problem Tutorial: "吉良吉影的奇妙计划"

问题可以看成是找到有多少长度为$2n$的$-1,0,+1$组成的序列，满足：

0总是成对出现

不包含4个连续0

$+1,-1$不能相邻

所有数的和为0

现在补充几个在求答案中需要用到的条件：

1.序列的前缀和总是非负

2.序列的前缀和总是为正（除了整个序列的和为0）

3.序列的结尾是0

4.序列的结尾非0

现在用$f_{ab\cdots i}$来表示满足额外条件$a,b,\cdots$的长度为$2i$的序列数，用S_i表示序列前i位（序列位置从1开始标号）的和。目标即求f_n。

那么有

$f_i = f_{3i} + f_{4i}$

$f_{1i} = f_{13i} + f_{14i}$

$f_{13i} = f_{14i-1}$ （因为不能包含4个连续0，对于所有满足条件3的序列去掉末尾的两个0，就会变成满足条件4的序列）

$f_{14i} = \sum_{j=0}^{i-1} f_{13j} f_{2i-j}$ （对于每个满足条件14的序列，总可以找到最大的一个$x < 2n$满足$S_x = 0$，容易证明x必然为偶数，对于任意的$y(x < y < 2n)$有$S_y = S_y - S_x > 0$，即可以将原序列看成满足条件13的长为x的序列（这里其实应该是满足条件1且末尾不为-1的，而由于满足条件1末尾不可能为$+1$，所以等价于满足条件13）与满足条件2的长为$2n - x$的序列（开头必为$+1$，所以前半段末尾不能为-1）拼接而成）

$f_{2i} = f_{1i-1}$ （满足条件2的序列的开头必然是$+1$，结尾必然是-1，将开头与结尾去掉后，满足条件1）

$f_{3i} = f_{4i-1}$

$f_{4i} = \sum_{j=0}^{i-1} 2 f_{3j} f_{2i-j} + f_{4j} f_{2i-j}$ （类似求f_{14i}）

接下来只需要将式子化简使用两次分治fft即可求出f_n。

Problem Tutorial: “Souls-like Game”

长度为n的线段树的区间长度只有$O(\log n)$种，预处理矩阵的线段树区间次幂，打标记时直接使用预处理出来的值而不是每次重新计算。

也可以不对任意的n证明上述结论，把线段树的长度扩展到2的幂次，直接根据线段树的结构就可以得出只需要预处理$\log n$个结论。

时间复杂度$O((n + qlogn) \times 3^3)$。

Problem Tutorial: “地球上最漫长的一天”

有一个明显的贪心思路是一直连续取到交集为空之后，开始下一个分组。

先考虑如何判断n个圆的交集是否为空。

二分交点的横坐标x，计算所有圆在x上的纵坐标的区间。如果有这些区间有交点，那么交集不为空；否则拿出两个交为空的区间对应的圆，如果它们没有交点，那么交集必然为空，否则交点必然全部在x的一侧，缩小交点的可行区间。二分个60次如果还没有找到交点就说明交集极大概率为空了。

从第1个圆开始倍增有交集的连续区间长度，当把区间长度限制在$[2^k, 2^{k+1}]$之间时，用这个区间作为初始区间去二分答案。下一个起始位置也这样做。总时间复杂度$O(nlogn60)$。

Problem Tutorial: “穗乃果的考试”

做法1：

问题可以转化为 $O((nm)^2)$ 枚举两个 1，统计有多少个矩阵同时包含这两个 1，然后把这 $O((nm)^2)$ 个矩阵数目全部累加起来即可。

实际上可以先 $O(nm)$ 枚举每个 1，第二个 1 的信息总和可以通过预处理二位前缀和 $O(1)$ 求出来。

做法2：

问题相当于定义一个矩阵的权值为其中1的个数的平方，然后求权值和。记$sum_{x,y}$为二维前缀和，则相当于求$\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=0}^{i-1} \sum_{l=0}^{j-1} (sum_{i,j} - sum_{k,j} - sum_{i,l} + sum_{k,l})^2$，拆开之后统计。

Problem Tutorial: “二人的白皇”

考虑点分治。对于一条询问(u,v)，如果它没有跨过了当前点分治的分治重心，递归到相应的子树；如果它跨过了分治重心，那么除了第一次之外，之后再跨过分治重心p，一定至少有一端是到某个叶子节点且这个叶子节点和之前的某个分治中心相邻。如果我们预处理可能出现的这样的链所在的子树中所有节点与这条链的距离，那么此时我们就可以$O(1)$求出这半段对应的答案，剩下的半段继续递归到对应的子树里面分治。查询时间复杂度为$O(q \log n)$。

这样预处理的时间复杂度为所有需要预处理的链的所在子树的大小之和，事实上这个大小是$O(n \log n)$的。考虑点u作为更高节点的可能性，每一层分治的时候递归下去的子树大小至少减半，

而每一层只会有一条端点与u相邻且需要被预处理的链，所以u作为更高端点的预处理时间复杂度大小是$O(size(u))$的，所以总时间复杂度为$O(n \log n)$。

Problem Tutorial: "岸边露伴的人生经验"

把一个点p看做一个20位的二进制数n 对于第i维： 若p[i] == 0, n[2 * i, 2 * i + 1] = 00 若p[i] == 1, n[2 * i, 2 * i + 1] = 01 若p[i] == 2, n[2 * i, 2 * i + 1] = 10

考虑计算p1与p2的距离，那么计算n1 xor n2， 根据[2 * i, 2 * i + 1]的取值可以还原出p1, p2在第i维的差的绝对值，因此我们可以用n1 xor n2计算dist(p1, p2)

因此利用FWT计算出所有 nx = n1 xor n2 成立的次数，再将nx转换为距离，统计即可

Problem Tutorial: "去音乐会"

首先假设b <= d，考虑一个简单版本：b - a = 1时，不考虑边界情况，问题可以重新表述成"计算满足l <= d * x mod m <= r 且 0 <= x < n的整数x的个数"，这可以转换成一个普通的直线下整点计数问题: lattice_count(n, m - l, d, m) - lattice_count(n, m - (r + 1), d, m)；lattice_count(n, a, b, m) 表示 $\sum_{i=0}^{n} \lfloor ((a + b * i)/m) \rfloor$

前面的版本只需要计算d * x mod m有多少次落入[l,r]，现在来看b - a值任意的情况，我们发现d * x mod m的值对于答案的贡献是个分段一次函数（最多4段），斜率为1/-1/0三种，所以现在我们只要求出d * x mod m落入[l,r]的值的和是多少，就可以求出每段的答案和

考虑直线下整点计数问题中我们找出来的点集，每个点(x/m, y/m)对应的原问题中d * x mod m的值为 y - (b * x + a)，于是求d * x mod m的和只需求出直线下整点点集的x坐标和及y坐标和即可，这个问题对直线下整点计数的类欧算法做些修改就可以做了

所谓对类欧算法的修改就是完全用几何意义表示直线下整点计数，包括维护当前坐标系与原始坐标系的对应关系以便快速求出原始坐标系下的点集和

单组时间复杂度$O(\log(b * d))$