1001

首先，我们通过一些树上的操作找到两条链的相交链 S_x, T_x（通过 LCA 和一些分类讨论）

因为本题说了他们不会在边上相遇，考虑在点上相遇的情况即可。

由于本题的 n 较小,所以我们可以枚举相交链上的每一个点，然后计算他们在这个点最早相遇的时间，找到其中相遇时间最早的点作为答案输出。

对于相交链上的一个点x，我们可以计算出 A 到达 x 的时间满足 $2k_1 \cdot \mathrm{dis}(S_a, T_a) + \mathrm{dis}(S_a, x)$ 或者 $2k_1 \cdot \mathrm{dis}(S_a, T_a) + \mathrm{dis}(S_a, T_a) + \mathrm{dis}(T_a, x)$。（其中$k$为任意正整数）

类似的，B 到达 x 的时间满足 $2k_2 \cdot \mathrm{dis}(S_b, T_b) + \mathrm{dis}(S_b, x)$ 或者 $2k_2 \cdot \mathrm{dis}(S_b, T_b) + \mathrm{dis}(S_b, T_b) + \mathrm{dis}(T_b, x)$。然后两两联立成 4 个二元一次同余方程，使用拓展欧几里得算法求解最小正整数解即可。

时间复杂度 $O(nm \log n)$。

1002

树形动态规划

很抱歉题面里说的是$2 \leq n$，但是数据里有$n = 1$的情况。

定义状态 $\mathrm{dp}_{i,0}$ 表示 i 这个点目前没有路由器安放，i 也不在路由器的覆盖范围内，i 的子孙全部被路由器覆盖的最小代价。

$\mathrm{dp}_{i,1}$ 表示i这个点安放了路由器，i的子孙全部被路由器覆盖的最小代价。

$\mathrm{dp}_{i,2}$ 表示 i 这个点目前没有路由器安放，但是 i 在路由器的覆盖范围内（可以由它的一个儿子安放），i 的子孙全部被路由器覆盖的最小代价。

那么我们有转移方程，这里为了防止新的 dp 值覆盖了老的 dp 值，我们定义 DP 数组为考虑了子节点 k 的 dp 值，dp 数组为没有考虑子节点 k 的 dp 值。

$$\begin{aligned}
&\mathrm{DP}_{i,0} = \mathrm{dp}_{i,0} + \mathrm{dp}_{i,2} \\
&\mathrm{DP}_{i,1} = \min\{\mathrm{dp}_{i,0}, \mathrm{dp}_{k,1}, \mathrm{dp}_{k,2}\} \\
&\mathrm{DP}_{i,2} = \min\{\mathrm{dp}_{i,2} + \min\{\mathrm{dp}_{k,1}, \mathrm{dp}_{k,2}\}, \mathrm{dp}_{i,0} + \mathrm{dp}_{k,1}\}
\end{aligned}$$

时间复杂度 $O(n)$。

1003

设计状态为 $\mathrm{dp}_{L,R,x,k}$ 为我们用编号 $[L, R]$ 区间内的卡牌进行操作后只剩一张等级为 x ，种类为 k 的卡牌所能获得的最大收益。特别的，我们定义$\mathrm{dp}_{L,R,0,0}$为将编号 $[L, R]$ 区间内所有卡牌打出之后的最大收益。那么我们就有如下的转移式子

如果我们想要计算打出 $[L, R]$ 区间内所有卡牌所能获得的最大收益（也就是$\mathrm{dp}_{L,R,0,0}$），那么我们可以选择把他们拆分成两个子区间来解决（记为res_1），或者打出区间内仅剩的一张卡牌（记为res_2）。

$$\mathrm{res}_1 = \max_{i=L}^{R-1}\{\mathrm{dp}_{L,i,0,0} + \mathrm{dp}_{i+1,R,0,0}\}$$
$$\mathrm{res}_2 = \max_{x=1}^{R}\max_{k=1}^{m}\{\mathrm{dp}_{L,R,x,k} + \mathrm{val}(x,k)\}$$

$$\mathrm{dp}_{L,R,0,0} = \max\{\mathrm{res}_1, \mathrm{res}_2\}$$

如果我们想要计算 $[L, R]$ 区间内操作后只剩下一张卡牌的最大收益，如果这张卡牌是1级的，那么我们可以枚举这张卡牌的位置（假设为 i ），然后计算打出 $[L, i-1]$ ，$[i+1, R]$ 区间内的牌所能获得的最大收益。

$$\mathrm{dp}_{L,R,1,k} = \max_{i=L}^{i=R}\{\mathrm{dp}_{L,i-1,0,0} + \mathrm{dp}_{i+1,R,0,0}\} \quad \text{if } A_i = k$$

如果这张卡牌是高等级的，那么我们可以枚举左右区间的一个断点x，然后从 $[L, i]$ 得到一张低一级的卡牌，然后从$[i+1, R]$ 里得到另一张低一级的卡牌，合并得到答案。

$$\mathrm{dp}_{L,R,x,k} = \max\{\mathrm{dp}_{L,i,x-1,k} + \mathrm{dp}_{i+1,R,x-1,k}\} \quad \text{if } x > 1$$

最后的答案就是 $\mathrm{dp}_{1,n,0,0}$

复杂度是 $O(n^2(mR)^2)$ ，而且不难发现在$n \le 100$的情况下卡牌的等级的最大值只会有7。

1004

题意：给你一个凸多边形 A 和一个凸多边形 B 和一个圆 P($\text{以}X\text{为圆心，}R\text{为半径}$)，问你在圆 P 内随机选一个点 S ，问当凸多边形沿着向量$\vec{OS}$ 移动，得到凸多边形 A' ，问 A' 与 B 相交的概率有多大。（相交意味着存在一个点 w，满足$w \in A' \text{ and } w \in B$。

首先，我们通过闵可夫斯基和可以构造出顺着怎样的向量 $\vec{OS}$ 移动之后的 A' 会与 B 相撞，不难发现这些 S 构成了一个凸包（对闵可夫斯基不熟悉的话可以看一下这道入门题 https://www.luogu.com.cn/problem/P4557）。故题目就转化成这个凸包和圆 P 相交区域的面积，可以用三角剖分等方法。

最后输出的概率要注意一下不要输出-0.

有的队伍可能精度还有问题（唔验题的时候没有考虑到，可以优化一下自己的圆和多边形求交的板子，可以通过这道题来看看自己的板子精度是否合格http://acm.hdu.edu.cn/showproblem.php?pid=2892）

时间复杂度 $O(n)$.

1005

求出字符串的最小表示法，然后两个串如果循环同构那么两个字符串相等。利用hash判断即可。

利用后缀自动机或者后缀数组之类的做法也可以通过本题。

1006

考虑每次从叶子向根 dp，设 f_u 表示从 u 逃出的最小代价，记 s_u 表示从根节点到 u 的边权和，则有:

$$f_u = \min_v\{f_v + (s_u - s_v + L)^2\}$$

将式子展开后，并整理系数后可以得到:

$$\begin{aligned} f_i &= (s_j - s_i - L)^2 + f_j \\ &= s_j^2 + s_i^2 + L^2 - 2s_is_j - 2s_jL + 2s_iL + f_j \\ &= [(-2s_j + 2L)s_i + (s_j^2 - 2s_jL + f_j)] + (s_i^2 + L^2) \end{aligned}$$

观察形式可以发现对于每一个节点 i， 其余的节点 j 对于 i 的转移都可以写成形如 $k_jx_i + b_j$ 的形式。

这里我们可以利用李超线段树来维护每个点子树中的凸包，并通过线段树合并来继承儿子的答案。

李超树+线段树合并的复杂度可以分析出 $O(n\log V)$ 。

虽然线段树合并在树上每次都会执行直线的下放，但是注意到每一条直线深度不会超过 $O(\log V)$，所以复杂度为 $O(n\log V)$。

如果从根向叶子 dp 会更简单一点，直接用可持久化李超树，每一个点用父亲节点的状态计算即可，复杂度也是 $O(n\log n)$ 。

1007

设 k 元形式幂级数 $G(x_1,x_2,\cdots,x_n)$ 。

$$G(x_1,x_2,\cdots,x_n)=\sum_{i_1,i_2,\cdots,i_k} c_{i_1,i_2,\cdots,i_k}x_1^{i_1}x_1^{i_1}\cdots x_n^{i_n}$$

其中 $c_{i_1,i_2,\cdots,i_k}$ 表示向量 $(i_1,i_2,\cdots,i_k)$ 作为技能出现的次数。

我们希望求出 $F(x_1,x_2,\cdots,x_n)$ 表示:

$$F(x_1,x_2,\cdots,x_n)=\sum_{i=0}^{\infty}G(x_1,x_2,\cdots,x_n)^i=\frac{1}{1-G(x_1,x_2,\cdots,x_n)}$$

那么其中 $F(x_1,x_2,\cdots,x_n)$ 的系数 $d_{i_1,i_2,\cdots,i_k}$ 则表示利用技能集合凑出向量 $(i_1,i_2,\cdots,i_k)$ 的方案数。

对于存在障碍物的方案数求解我们只需要 $O(m^2)$ 的容斥即可，设 f_i 表示到达第 i 个点的方案数。

$$f_i=\text{ways}(p_i)-\sum_{j,p_j\le p_i}\text{ways}(p_i-p_j)f_j$$

这里 p_i 表示第 i 个关键点的坐标向量，$\text{ways}(p_i)$ 表示组合出向量 p_i 的方案数，定义 k 维向量的 p_j 小于 p_i 为每一维均小于，同时 p_i-p_j 为每一维相减。

现在我们只需要解决一个问题就是 k 元形式幂级数的求逆。

首先解决 k 元多项式乘法，对于常见的 k 比较小的情况可以将每一维的值域扩张成两倍，然后直接做卷积并舍弃进位的情况即可。

对于 k 较大的情况，我们多引入一维 t , 并引入占位函数 $\chi(i)=\left\lfloor\frac{i}{n_1}\right\rfloor+\left\lfloor\frac{i}{n_1n_2}\right\rfloor+\cdots+\left\lfloor\frac{i}{n_1\cdots n_{k-1}}\right\rfloor$ ，将函数表示成 $\sum_i f_ix^it^{\chi(i)}$ 。

多引入一个元可以增强对卷积的限制，我们原先的问题在于我们将向量重标号后，进行加法之后的标号会产生进位并对答案产生影响，在引入 t 之后还需要满足 $\chi(i)+\chi(j)=\chi(i+j)$ ，注意到有 $\left\lfloor\frac{i}{x}\right\rfloor+\left\lfloor\frac{j}{x}\right\rfloor-\left\lfloor\frac{i+j}{x}\right\rfloor\in\{-1,0\}$ ，所以 $\{\chi(i-j)+\chi(j)\}$ 合法的集合很小。于是我们只需要计算 $\sum_i f_ix^it^{\chi(i)}$ 在 $\bmod\left(t^k-1\right)$ 意义下的多项式乘法，也就是循环卷积。

由于 $k\le\log_2 N$ ，那么就可以对每一维度做长度为 $2N$ 的 DFT，然后对于 t 暴力做卷积。

对于求逆，只需像一元多项式一样利用牛顿迭代求解即可。求导也是一致的。

时间复杂度为 $O(kN\log N)$

1008

观察发现加体力的道具收益和休息是一样的，加属性的道具收益不如训练，增加训练效果的道具收益和训练一样。

得到有体力就训练，没体力就休息的决策，于是回合数n为偶数时属性值直接$(n/2) * 15$。

考虑回合数n为奇数的情况，训练休息轮流的策略会导致一个回合没有收益，考虑通过比赛和道具来获得这一个回合的收益。

首先需要通过一个加50的体力药或者两个加25的小体力药来获得比赛消耗的体力。相当于消耗一个回合但不消耗体力，获得50商店币，将多余的一个回合转化为了商店币。

然后考虑通过商店币能获得的属性，50商店币能转化为一次加6的属性书或者两次加3的小属性书。

所以n为奇数时被分为：浪费一个回合，属性多加3，属性多加6，属性多加7，一共四种情况，分别计算出对应的概率即可。

1009

鸽巢原理的应用，至少有一个堆中有 $\lfloor \frac{m-1}{n} \rfloor + 1$ 个元素。

判断 d 和 $\lfloor \frac{m-1}{n} \rfloor + 1$ 的大小关系即可。

1010

考虑绝对值拆开维护，当 $a_i < x$ 时，$a_i = x - a_i$，当 $a_i > x$ 时，$a_i = a_i - x$。

对于 $a_i \geq x$ 的情况，直接通过线段树维护。

对于 $a_i < x$ 的情况，相当于将 a_i 取反并且加上 x。考虑在线段树上维护这个东西，在一直处于 $a_i < x$ 的情况下，每次操作会将 $x_1 - x_2 + x_3 - ... + x_{j-1} - a_i$ 变化为 $-x_1 + x_2 - x_3 - ... - x_{j-1} + x_j + a_i$,这时可以发现 a_i 的数值不发生变化，只有符号改变, x 标记可以直接叠加维护，所以也可以通过线段树维护。

根据 $x_{j-1} \leq x_j$ 的性质，发现如果 $a_i < x_{j-1}$ ，修改后的 $|a_i - x_{j-1}| \leq x_{j-1} \leq x_j$ ，所以从 $a_i \geq x$ 转变为 $a_i < x$ 的情况后就不会改变，所以转变总的次数是有限的。

直接将数分为两类，分别通过线段树维护，在 $a_i \geq x$ 转变为 $a_i < x$ 时暴力的从一棵树转移到另一棵树上。

因为暴力修改的次数是有限的，所以总的复杂度为$O(n \log n + m \log n)$。

1011

和1010的操作是类似的，但是去掉了 x 不递减的性质之后对于 a_i 取反的操作标记不能叠加，所以难以直接通过线段树维护。

考虑分块，对于数的标号分块，每一个块建一个平衡树进行维护，平衡树的关键字为数的大小。

对于整块的操作，可以将平衡树分为小于 x 和大于等于 x ，对于小于的部分进行翻转和加操作，对于大于等于的部分直接进行加操作。因为在 $a_l < a_r$ 时，$x - a_l < x \leq x + a_r$，所以仍然满足平衡树的要求。

对于零散块直接重构整颗平衡树。

此时的复杂度为$O(m\lfloor\frac{n}{q}\rfloor\log q+mq\log q)$,其中 q 为块的大小，为 $\sqrt{n}$ 时总复杂度为 $O(m\sqrt{n}\log\sqrt{n})$。

考虑进一步优化，发现对于零散块，没有进行修改的数显然大小关系不变，对于修改的数可以分为小于 x 和大于等于 x 两类，两类内部的大小关系也不变，所以可以将修改后的零散块分为三个分别有序的序列。

显然可以通过 $O(n)$ 的做法合并这三个有序的序列，得到修改后的按数的大小排列的序列，再通过 $O(n)$ 的建树得到重构后的平衡树。这时我们得到了一个 $O(q)$ 的零散块做法，q 为块的大小。

因为零散块的做法优于整块，考虑对于块的大小进行平衡，可以得到在块大小为 $\sqrt{n\log n}$ 近似最优，此时总的复杂度大约为 $O(m\sqrt{n\log n})$

如果你的常数足够优秀，或许不需要进行零散块的优化和块大小的调整也能通过此题。

1012

考虑每棵树的 SG 函数，可以发现

$$\mathrm{SG}(u)=(\mathrm{SG}(v_1)+1)\ \mathrm{xor}\ (\mathrm{SG}(v_2)+1)\ \mathrm{xor}\ \cdots(\mathrm{SG}(v_l)+1)$$

先计算出一个根节点的 SG ，比如先计算出1为根时，各个子树的 SG 和 1 自己的 SG ,然后通过换根的方法得到所有节点的SG 函数。

$$\mathrm{SG}(v)=(\mathrm{SG}'(v))\ \mathrm{xor}\ ((\mathrm{SG}(u)\ \mathrm{xor}\ (\mathrm{SG}'(v)+1)+1)$$

其中，$\mathrm{SG}(u)$ 表示 u 的 SG 函数，$\mathrm{SG}'(u)$ 表示在子树 u 的 SG 函数。

最后计算 $\frac{1}{n}\sum[\mathrm{SG}(u)>0]$ 输出即可。