# A. 能量采集

每当容器中有 $K$ 个能量 A 时，就会对答案加一，所以我们可以计算每一处位置的贡献。

设 $f(i,j,p)$ 表示当前人走到格子 $(i,j)$，目前队列中最后一段有连续 $p$ 个能量 A 的方案数。转移方程很简单：

$$f(i,j,0)=\binom{i+j-2}{i-1}$$

$$f(i,j,p+1)=f(i-1,j,p)+f(i,j-1,p),\text{if } s(i,j)=\text{A}$$

我们对所有格子都计算出 $f(i,j,K)$，但此时这个人只走到了 $(i,j)$，那么方案数一共有 $f(i,j,K)\cdot\binom{n-1-(i-1)+m-1-(j-1)}{n-1-(i-1)}$。最后将所有格子的贡献全部加起来即可。由于是计算期望，所以最后答案还需要除以 $\binom{n+m-2}{n-1}$。

时间复杂度：$\mathcal{O}(nmK)$

# B

考虑移动的过程，比如我们往上移动，通过了某个 $a_k$，那么再往上就只用考虑所有 $a_j>a_k$ 的 $j$。

考虑按照最大值分治建树，即建立一个笛卡尔树，那么我们移动的过程显然就变成了在树上不断往上走，每走到一个点，这个点的所有子树就一定都可以访问，设 $Sb_x=\sum_{y\in subtree(x)}b_y$，那么当前在 $x$，能走到 $x$ 的父亲的条件就是 $a_{fa_x}-sb_x\le v$，其中 $v$ 是初始的强度。

所以建立笛卡尔树后，对于每次询问，只需要通过树上倍增或者树链剖分等方法维护，找到第一个大于 $v$ 的边，那么边的儿子的 $Sb+v$ 就是答案

复杂度 $O(nlogn)$

# C. 往日重现

由于圆之间不会相交或相切，因此圆与圆之间的包含关系可以看成一棵树，查询的是这两个点所属的最小圆在树上的距离。对于这些圆和查询点，考虑对 $x$ 轴做扫描线，将圆分为上下半圆维护包含关系。当扫描线扫到某个圆的左边界（即 $x-r$ 位置）时，将上半圆和下半圆加入平衡树中，并查询上半圆上方最近的半圆是哪一个。如果是某个上半圆，则证明这个圆被上半圆所在圆包含，置这个圆的父亲为这个上半圆所在圆，如果是下半圆，则证明这个圆和下半圆所在圆都被另一个大圆包含，置这个圆的父亲为这个下半圆所在圆的父亲。对于查询点，同样利用扫描线维护包含关系。当扫描线扫到某个圆的右边界（即 $x+r$ 位置）时，在平衡树上删除这两个半圆。这样就构建出了一棵树和一系列树上距离查询，使用 RMQ-LCA 即可处理这些查询。

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

# D.小美爱画鱼

由于坐标范围只有$1\times10^5$，总共只可能在 $2\times10^5$条对角线上出现,因此我们可以考虑维护统计每个对角线上的线段的起点和终点。

判断是否重复覆盖可以用差分的方式处理，即将每个起点处+1，终点处-1，查看计算过程中是否出现大于1的位置，若有即为有重复覆盖。

复杂度$O(Tn\log n)$。

# E.小团来打字

直接考虑模拟即可。

首先将相连的相同的字符合并。

对于每个相同的串，其会输入的次数为$\lfloor \frac{x-1}{k} \rfloor \times 2k + (x - \lfloor \frac{x-1}{k} \rfloor \times k)$次。

复杂度为 $O(n \log n)$或$O(n)$。

# F.炸弹鸭

我们可以考虑将询问离线，这样我们可以将问题转化为，加入所有大小小于等于k的边后，有多少个点的度数大于d。

因此我们可以一个后缀和来表示度数大于等于当前值的点有多少个，采取双指针的思想可以做到$O（n）$的复杂度实现，也可以使用树状数组来维护后缀和等。

复杂度为$O((m+q)\log n)$。

# G

考虑从所有重要城市为源点开始多源 BFS。记 $d_u$ 为 $u$ 到最近的重要城市的距离。当 BFS 第一次访问到一个点 $v$ 时，设此时是从 $u$ 点访问 $v$ 点，则赋值 $d_v := d_u + 1$，若不是第一次访问到，则说明 $d_v \leq d_u + 1$。进而，若 $d_v = d_u + 1$，则说明经过 $u$ 点到达 $v$ 点的这一部分重要城市与 $v$ 的距离也是最小距离，于是距离 $v$ 点为最短距离的重要城市个数需要增加上来自 $u$ 点的这一批。于是我们可以通过以下伪代码求出 $d$ 和第一问的答案：

```
q = queue()
d = [-1] * n
cnt_t = [0] * n
for c in important_cities:
    d[c] = 0
    cnt_t[c] = 1
    q.push(c)
while not q.empty():
    u = q.front()
    q.pop()
    for (u, v) in E:
        if d[v] == -1:
            d[v] = d[u] + 1
            q.push(v)
        if d[v] == d[u] + 1:
            cnt_t[v] = cnt_t[v] + cnt_t[u]
```

此处 $cnt_t[u]$ 即在 $u$ 点的军队所保护的重要城市的数量。

现在考虑如何求每个重要城市被多少个军队所保护。现在每个城市到重要城市的距离都已经被求出。假设 $u$ 城市有一个军队，则只要距离 $u$ 为 $\leq d_u$ 的所有重要城市都被 $u$ 城市的军队保护（因为根本不存在距离 $< d_u$ 的重要城市）。

我们可以这样考虑：因为只要距离 $u$ 为 $\leq d_u$ 的所有重要城市都被 $u$ 城市的军队保护，所以我们可以从 $u$ 开始走 $d_u$ 步，只要能走到的位置都是可以被保护的。假设有一条边 $(u, v)$，若 $d_v = d_u - 1$，那么向 $v$ 走一步是离重要城市更近一步的；若 $d_v \geq d_u$，则说明一旦走到了 $v$，从 $v$ 再走 $d_u - 1$ 步无论如何不可能走到重要城市。因此，我们只需要按照 $d_u$ 从大到小的顺序枚举所有 $u$，然后对于所有与 $u$ 直

接相连的点 $v$，若 $d_v = d_u - 1$，则当前能走到 $u$ 的所有军队都可以继续往 $v$ 走，即 $cnt_t[v] := cnt_t[v] + cnt_t[u]$。伪代码如下：

```
bucket = [[]] * n
cnt_c = [0] * n
for t in troops:
    cnt_c[t] = 1
for u in V:
    bucket[d[u]].append(u)
for i = n - 1 to 1:
    for u in bucket[i]:
        for (u, v) in E:
            if d[v] == d[u] - 1:
                cnt_c[v] += cnt_c[u]
```

此处 $cnt_c[u]$ 即保护重要城市 $u$ 的军队的数量。

至此，我们在 $O(n)$ 时间复杂度内求出了答案。

# H. 约瑟夫问题

本题做法比较多，这里提供线段树做法。

使用线段树来维护每一个位置是否还有人在，有的话当前位置的值是 1，否则就是 0。对于输入的 $k$，首先对当前剩余人数取余之后就是最后一个周期还需要报的人数。线段树维护区间和，只需要在线段树上做二分即可。

当然，本题时限给得比较足够，可以最后使用二分+线段树，或者整个题使用分块的做法也行。

时间复杂度：$\mathcal{O}(n \log n)$ 或 $\mathcal{O}(n \log^2 n)$ 或 $\mathcal{O}(nB)$，其中 $B$ 表示分块的块大小。

# I

考虑将 $9y$ 移动到右边，变成 $8x + 10z = n - 9y$，之后讨论 $n$ 的奇偶性，变成 $4x + 5z = \frac{n-9y}{2}$，考虑对于每个合法的 $y$，就是一个二元一次方程，可以用扩展欧几里得求出 $x$ 和 $z$ 的通解。通解的表示形式可以看成是一条直线上的整点数，这个可以求区间和。对于所有合法的 $y$ 求对应的解的个数，是一个形如 $\sum \lfloor \frac{ax+b}{c} \rfloor$ 的形式，可以用类欧几里得来求。

时间复杂度：$\mathcal{O}(T \log n)$

当然，由于本题方程形式比较特殊，可以用其他方法通过本题。

# J. 数矩形

考虑可以通过对角线平分且相等来判定四边形是否为矩形，因此考虑 $\mathcal{O}(n^2)$ 地枚举矩形对角线，记录中点和对角线长度。对于每个中点，两条不同且长度相等的对角线可以组成矩形。考虑统计二元组 $(P, d)$，其中 $P$ 为对角线中点，$d$ 为对角线长度的数量，如果这个二元组有 $x$ 个，那么会形成 $\frac{x(x-1)}{2}$ 个矩形。统计过程可以使用 map 或直接排序。本题实现中全部使用整数，浮点数可能产生误差，需要考虑。

时间复杂度：$\mathcal{O}(n^2 \log n)$，空间复杂度：$\mathcal{O}(n^2)$。

# K. 打地鼠

考虑对于 $m \geq n$ 的图，一定存在一个环，那么地鼠一定存在一个绕环游走的方案躲避 Kanade，所以此时 Kanade 必输，输出 `Lose`。

而由于保证图的连通性，则有解的情况一定为 $m = n - 1$，也就是一棵树，因此可以在树上模拟这个击打序列，记录一个 `bool` 数组表示这一轮地鼠可能出现的位置，一轮击打后，更新下一轮地鼠可能出现的位置即可。一轮更新相当于遍历所有边，复杂度为 $\mathcal{O}(n)$，直接这样模拟复杂度可以接受。

如果不考虑 $m$ 与 $n$ 的关系，还有一种利用 bitset 使时间复杂度达到 $\mathcal{O}(\frac{n^2k}{w})$ 的模拟方法，但由于时限过紧不保证可以通过本题。

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

# L. 树边重排

根据题目要求，可以考虑从节点 $n$ 开始 dfs 遍历整棵树，记录每个点的父节点即可。

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

# M

首先，$x_i \leq t$的条件可以去除

考虑直接定义$i$进程在$t$时刻的响应比为$\frac{t-x_i}{s_i}$

不符合条件的点有$x_i > t$，此时响应比为负数

而符合条件的点响应比都为正数，因此去除条件不会影响求响应比最大的进程

同时，若求得最大响应比为负数，则说明没有满足条件的进程，输出$-1$

简化后，问题就变成了对许多$t$，求$\frac{t-x_i}{s_i}$的最大值是多少

考虑在任意时刻$t$，假设有进程$i, j$

若进程$i$的响应比要比$j$高，一定有：

$$\frac{t-x_i}{s_i} > \frac{t-x_j}{s_j} \iff t \cdot (\frac{1}{s_i} - \frac{1}{s_j}) > \frac{x_i}{s_i} - \frac{x_j}{s_j}$$

若$1/s_i > 1/s_j$，那么$i$优于$j$的充要条件是：

$$t > \frac{\frac{x_i}{s_i} - \frac{x_j}{s_j}}{\frac{1}{s_i} - \frac{1}{s_j}}$$

将每个进程转换为一个点$(1/s_i, x_i/s_i)$，那么该充要条件转换为$i$的点在$j$的点右边，且两点间的斜率小于$t$

简单来说，若两点间斜率小于$t$，则右边的点更优，若两点间斜率大于$t$，则左边的点更优，这是一个典型的凸包

于是，对$(1/s_i, x_i/s_i)$做凸包，只有凸包上的点才可能是答案

每个询问转换为求斜率为$t$的直线与凸包的切点的响应比，输出该响应比即可

做凸包$O(nlogn)$，处理询问$O(nlogn)$，总复杂度$O(nlogn)$

# N. 本质不同的 01 环计数

所有长度为$k$的子段和相同，考虑两个相邻的子段
$a_i + a_{(i+1)\%n} + \ldots + a_{(i+k-1)\%n} = a_{(i+1)\%n} + a_{(i+2)\%n} + \ldots + a_{(i+k)\%n}$，即对于所有$a_i = a_{(i+k)\%n}$，这等价于环的循环节为$gcd(n, k)$，相当于我们要求所有本质不同的长度为$gcd(n, k)$的01环方案数。

考虑枚举最小循环节长度恰好为$i$的方01串方案数，相当于$2^i$减去所有因子长度的方案数，而最小循环节长度恰好为$i$的方案数正好是本质不同的01环的方案数的$i$倍。

再考虑所有循环节为$i$的01环的本质不同的方案数，就等于所有最小循环节是$i$的因子的本质不同的方案数的和。

前面这两个都可以通过埃氏筛以调和级数的复杂度来完成，之后询问只要求$gcd(n,k)$就可以直接得到答案了。

同样，如果有群论的知识，可以直接使用burnside数不动点来解决这个问题。

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

# O

首先根据贪心思想，一定在越高的层付出代价返回更优，因为返回后打败之前的敌人获得的等级更多。
设 $s_1,\cdots,s_n$ 表示 $a_1,\cdots a_n$ 的前缀和，从低层到高层模拟，如果无法通过当前这层 $i$ ，那么返回可以得到的等级数为 $s_{i-1}+1$，做除法计算通过需要返回的次数，并加上即可。
复杂度 $O(n)$

然后实际造数据的时候疏忽了，把暴力计算次数的做法放过去了，红多尼私密马赛（鞠躬）。

# P. 三维模型

本题中，我们可以把三角形和三角形的每条边抽象为点，令点 $i\ (1\le i\le n)$ 表示第 $i$ 个三角形，点 $n+3\times(i-1)+j\ (1\le i\le n,1\le j\le 3)$ 表示第 $i$ 个三角形中的第 $j$ 条边，如果某条边属于某个三角形，就建立一条这个三角形代表的点与这条边代表的点的边。由于两个三角形可以共用同一条边，这意味着三角形边的一些点是可以合并起来看待的，所以考虑用 map 维护边，当初次插入 map 时为这条边新建一个点，如果 map 中查询到此边已经存在一个表示它的点，那么之后就用这个点与三角形代表的点连边。需要注意我们用一个无序点对代表边，插入 map 前需转化成有序点对。

这样可以构建一个三角形的点与边之间关系的图。可以发现同属于一个三维模型的三角形，所代表的点在这张图内一定是连通的，也就是说，图中的一个连通块代表一个三维模型。所以考虑遍历图中的连通块，统计其中的三角形个数即可。在图上使用 dfs 或 bfs 均可实现这一操作。

同样，也可以使用并查集来实现这个操作，也就是把同属于一个三维模型的三角形和边放在同一个集合中，最后统计集合个数和每个集合中三角形个数即可。

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

# Q

最小和最大一定是 $1,2*n-1$

最小：考虑如果某个 $l_i=1,r_i\neq 2$，那么一定有一个操作 $l_j=2$，先进行 $i$，再进行 $j$，那么 $a_1=i,a_2=j$，答案为 $1$。如果 $l=1,r=2$，那么没有别的 $l_j=2$，$a_1=i,a_2=0$，答案为 $1$

最大：讨论类似，如果 $l_i=2*n-1,r_i=2*n$，答案为 $2*n-1$。若 $l=2*n-1,r=2*n$，答案类似为 $2*n-1$

复杂度 $O(1)$

# R

枚举所有情况即可