# Tutorial for Problem A. 解方程

注意到当 $\sqrt{n}$ 是无理数时，方程可化为

$$\begin{cases} x = y + z \\ n = 4yz \\ x \geq \sqrt{n} \\ 0 \leq y \leq z \end{cases}.$$

上述方程组有自然数解时 $n$ 必然是 4 的倍数，此时解数为 $m = \frac{n}{4}$ 约数个数的一半，对应的解为

$$\left\{ \left( y + \frac{m}{y},\, y,\, \frac{m}{y} \right) \middle| \, 1 \leq y \leq \sqrt{m},\, y \text{ is a divisor of } m \right\}.$$

此处由于第一条限制和第四条限制同时满足时，第三条限制必然满足，故可以不考虑第三条限制。

而当 $q = \sqrt{n}$ 是有理数时，方程可化为 $\sqrt{x - q} = \sqrt{z} - \sqrt{y}$，取 $x = q + z,\, y = 0$ 时有无穷多组自然数解。实际的解数会更多，对于这种情况直接判定解数无限多即可。

# Tutorial for Problem B. 旅途

若小象还没有游玩到所有的城市，则小象游玩的城市可以看成环上的一个区间。

我们只关心这个区间的长度和小象在这个区间里的位置，因此可以令 $dp(i, x, y)$ 表示在小象第 $i$ 天游玩的城市顺时针方向有 $x$ 个已经游玩的城市、逆时针方向有 $y$ 个已经游玩的城市对应的所有局面的概率之和，转移时只需要枚举第 $(i+1)$ 天的城市和第 $i$ 天的城市的相对位置。

由上述动态规划可以得到 $f(1), f(2), \ldots, f(n-1)$，由 $\sum_{i=1}^{n} f(i) = 1$ 可求出 $f(n)$，至此问题解决。

上述做法的时间复杂度为 $\mathcal{O}(N^2 m + Nk)$，这里 $N = \min\{n, m\}$。实际的常数很小，跑得很快。

顺带一提，此题可以 $\mathcal{O}(Nm \log N + N \log k)$ 解决，做法留给读者思考。

# Tutorial for Problem C. 项链与计数

答案即每个边双连通分量里两两可达的点数。

首先利用 Kruskal 算法得到一个生成森林，并且尽量使用编号较小的边，这使得每次合并边双连通分量时只会影响一条当前树上已有的边。

然后利用并查集合并树上已有的边连接的不同边双连通分量，同时维护边双连通分量里的点数即可。

具体来说，为了便于实现，可以将生成森林中每棵树建成有根树，对每个边双连通分量维护离根最近的点，则每次合并一系列分量时可以只考虑离根最近的点与其父节点，合并对应的两个边双连通分量。

这样使得每次可以不用考虑一条路径上所有边，而是只考虑产生合并效果的边，从而做到时间复杂度 $\mathcal{O}(n\alpha(n) + m)$。注意，若维护并查集合并时不按秩合并，时间复杂度可能变为 $\mathcal{O}(n \log n + m)$，可能会超时（也可能不会）。

由于输入数据较大（小于 29 MiB），程序运行的主要耗时应该在读入部分，所以时限开得较大。

# Tutorial for Problem D. 战术安排

不妨考虑枚举做题的顺序，则问题可以化为在三个流水线上安排工作的问题。注意到流水线的长度很小（只有 180），可以尝试用动态规划求解。

具体来说，令 $dp(i,p,q)$ 表示只考虑前 $i$ 个任务使得前两个流水线上工作结束时间为 $p$, $q$ 时，第三个流水线上最早工作结束的时间，转移时只需要模拟任务分配的方式即可。此外，这里动态规划只用考虑 $0 \le p,q,dp(i,p,q) \le 180$ 的信息，可行的转移数量会进一步减少。但实际上，直接枚举所有可能的顺序（共 720 种）较慢，很难通过此题。

不妨尝试利用搜索配合剪枝的技巧优化枚举做题顺序的过程，或者改进动态规划的模型，直接计算 $dp(S,p,q)$ 表示只考虑属于集合 $S$（共 64 种）的任务时对应的信息。这两种改进做法的效率差别不大，都可以通过此题。

# Tutorial for Problem E. 最长上升子序列

若修改前整个序列的最长上升子序列长度是 $m$，则修改后的最大值只有 $m$ 和 $(m+1)$ 两种可能。

首先考虑修改后长度变为 $(m+1)$ 的情况。在这种情况中，修改后 $a_x$ 必然属于最长上升子序列中。令 $pre(i,j)$ 表示在 $a_1$, $a_2$, ..., $a_i$ 中满足子序列末尾元素小于 $j$ 的最长上升子序列长度，相应地，定义 $suf(i,j)$ 为 $a_i, a_{i+1}, \ldots, a_n$ 中满足子序列开头元素大于 $j$ 的最长上升子序列长度，则这种情况对第一问的贡献为 $\max\limits_{y\ge 0}\{pre(x-1,y)+suf(x+1,y)\}$，而第二问所求最小的 $y$ 一定是 0 或者某个 $(a_i+1)$，因此只需要考虑 $\mathcal{O}(n)$ 种可能的 $y$ 对应的 $pre(x-1,y)+suf(x+1,y)$。

考虑 $pre(i)$ 与 $pre(i-1)$ 的差异，不难注意到 $pre(i)$ 是分段函数，且每一段函数值相等，不同段函数值之差恰好是 1。不妨令前 $i$ 个元素里能达到长度 $k$ 的上升子序列里最小的末尾元素是 $low(i,k)$，则 $low(i)$ 是一个严格递增的序列，同时它也对应了 $pre(i)$ 里的分段点，而 $low(i-1)$ 和 $low(i)$ 只有至多一个信息的差异，因此 $pre(i)$ 和 $pre(i-1)$ 的差异仅有一段区间。预处理出这些区间后即可枚举 $x$ 并用线段树维护出 $pre(x-1,y)+suf(x+1,y)$ 关于 $y$ 的区间最大值，以及这样的最大值对应的最小非负整数 $y$。

然后考虑修改后长度仍为 $m$ 的情况。这里需要根据修改后 $a_x$ 是否参与最长上升子序列的构成分两种情况考虑，其中参与构成的情况已经在之前的分析中解决，现在需要解决的是不参与构成的情况。若修改前 $a_x$ 可以不属于最长上升子序列，则修改后也可以不属于。否则，为了保证最长上升子序列的长度不减小，修改后 $a_x$ 也必须属于最长上升子序列。因此这种情况可以转化为判定修改前 $a_x$ 是否必定属于最长上升子序列。

考虑能够属于最长上升子序列里的元素满足的性质。令 $f(i)$ 表示以 $a_i$ 结尾的最长上升子序列长度，$g(i)$ 表示以 $a_i$ 开头的最长上升子序列长度，只有当 $f(i)+g(i)-1=m$ 时，$a_i$ 才属于至少一种可能的最长上升子序列，否则一定不存在最长上升子序列包含 $a_i$。

找出这些满足性质的位置组成的集合 $S$ 后，只需要对于任意的 $i,j\in S$, $i<j$, $a_i<a_j$, $f(i)+1=f(j)$，连一条 $i$ 到 $j$ 的有向边，那么任意一个最长上升子序列都可以用这个有向无环图中的一条极长路径表示。判断一个元素 $a_x$ 是否属于所有的最长上升子序列，相当于判断 $S$ 中是否不存在其他元素 $i$ 满足 $f(x)=f(i)$。

对于第二问来说，如果不是所有最长上升子序列都包含 $a_x$，则最小的 $y$ 可以是 0，否则 $y=\min\limits_{i\in S,i<x,a_i<a_x,f(i)+1=f(x)}\{a_i\}+1$。不难利用反证法证明，当 $x\in S$ 且 $a_x$ 是唯一一个 $f(x)$ 为某个定值的元素时，$S$ 中在 $x$ 左侧离它最近的位置便对应着元素 $\min\{a_i\}$。

综上所述，我们可以得出一个时间复杂度为 $\mathcal{O}(n \log n)$ 的做法。

由于最优的 $a_x = y$ 能够参与的最长上升子序列的长度只有两种，枚举这样的长度也是一种可行的做法。具体来说，枚举最长上升子序列中 $a_x$ 的前后元素 $a_i$ 和 $a_j$，它们必然满足 $f(i) + g(j) = m, m+1$，而对 $a_x$ 的影响是 $y$ 的最小值可以选 $(a_i + 1)$，因此只需要对每个 $a_i$ 计算出最右的 $j$，然后对 $i < x < j$ 的 $a_x$ 产生影响。这个做法可以不使用线段树，效率相对较好一点，不过时间复杂度也是 $\mathcal{O}(n \log n)$。

# Tutorial for Problem F. 木棍与多边形

我们可以枚举 $2^n$ 种可能的木棍组合能否凑成多边形，在能凑成的情况下使面积尽量大，然后再 $\mathcal{O}(3^n)$ 枚举子集的子集来合并信息，从而确定面积乘积的最大值。由于答案可能达到 $10^{24}$ 级别，为了避免较大的精度误差，可以选择精度更好的运算类型，也可以在合并信息时先对面积取对数来保证精度。

对于 $m$ 根木棍，令其长度为 $a_1, a_2, \ldots, a_m$，不难利用三角形不等式证明，这些木棍能组成一个简单多边形当且仅当 $2\max\{a_1, a_2, \ldots, a_m\} < \sum_{i=1}^{m} a_i$。

对于能够凑成多边形的情况，面积最大的多边形一定是凸多边形，并且总存在唯一的半径 $R$ 满足我们可以在一个半径为 $R$ 的圆周上选择 $m$ 个点构成 $m$ 边形，其边长依次为 $a_1, a_2, \ldots, a_m$。这个结论可以用调整法得到，不妨尝试画一个半径为 $\frac{1}{2}\max\{a_1, a_2, \ldots, a_m\}$ 的圆，将多边形的边依次对应到圆周上。若多边形覆盖的周长总和超过一个圆周，则意味着半径不够大，通过增大半径可以得到圆心在多边形内的圆满足条件；若周长总和没有达到一个圆周，也意味着半径不够大，但通过增大半径得到的圆对应的圆心会在多边形外部。

在这种构造多边形的方法里，多边形的面积等于圆的面积减去 $m$ 个弓型区域的面积。不妨留下这些弓型区域并尝试改变多边形的形状，通过这样的调整可以枚举到这个边长集合对应的所有可能的凸多边形形状。与此同时，圆被改变为周长固定的闭合图形，而在二维空间里，周长固定的闭合图形中面积最大的就是圆，因此多边形面积最大时一定能够画出这样的圆，使得多边形的点都在圆周上。那么剩下的工作便是二分确定唯一的半径 $R$，然后算出相应的多边形面积。

令所需的精度为 $\epsilon^{-1}$，则我们可以得到一个时间复杂度为 $\mathcal{O}(n2^n \log \epsilon + 3^n)$ 的做法，在实现上注意一些精度细节即可通过。