2022 国际大学生程序设计竞赛亚洲区域赛（南京站）

SUA 程序设计竞赛命题组

2022 年 12 月 18 日

I. 完美回文

题意

- 给定长度为 n 的字符串，令 $f(S, d)$ 表示将 S 左移 d 次后获得的字符串。若对于所有非负整数 d，$f(S, d)$ 都是回文串，则称 S 为完美回文。
- 每次操作可以修改一个字母，求将 A 变为完美回文的最少操作次数。
- $1 \leq n \leq 10^5$.

I. 完美回文

- 若 A 是完美回文，说明 $a_0 = a_1 = \cdots = a_{n-1}$。
- 因此枚举最终将 A 统一成哪个字符 c，我们需要将所有非 c 的字符都变成 c。答案就是 $\min(n - cnt_c)$，其中 cnt_c 是字符 c 在 A 中出现的次数。
- 复杂度 $\mathcal{O}(n + |\Sigma|)$，其中 $|\Sigma|$ 是字符集大小。

G. 邪恶铭刻

题意

- 维护一个序列，序列里一开始只有一个 1。您要处理 n 个事件，每次要么往序列里添加一个 1，要么从序列里拿出两个数，加起来再放回去，要么在这两种事件中选一个。
- 在每次都能从序列里拿出两个数的前提下，求序列平均值的最大值。
- $1 \leq n \leq 10^6$.

G. 邪恶铭刻

- 序列的平均值仅由序列中的元素和以及序列的元素数量决定。进行第一种事件后，答案的分子和分母都加 1（由于答案总是大于等于 1，这种操作会让答案不变或变小）；进行第二种事件后，答案的分母减 1（也就是答案变大了）。
- 显然我们要在合法的前提下，尽可能进行第二种事件。除非接下来马上就要变成非法状态，才考虑将之前一次自由选择从事件二反悔成事件一。
- 模拟这个贪心策略即可，复杂度 $\mathcal{O}(n)$。

D. 聊天程序

题意

- 给定一个长度为 n 的整数序列 $a_1, a_2, \cdots, a_n$，同时给定另外四个整数 k，m，c 与 d，可以进行以下操作至多一次：选择一个长度恰为 m 的连续子数组，并将一个长度为 m，首项为 c，公差为 d 的等差序列加到该连续子数组上。
- 最大化序列中第 k 大的值。
- $1 \leq k, m \leq n \leq 2 \times 10^5$.

D. 聊天程序

- 二分答案 x，将问题转化为“判断能否通过至多一次操作，使序列中第 k 大的元素大于等于 x”。
- 将大于等于 x 的数看成 1，小于 x 的数看成 0，此时问题变为“判断能否通过至多一次操作，使序列中 1 的数量大于等于 k”。

D. 聊天程序

- 接下来枚举操作位置，并计算进行操作后能否满足要求。考虑一个元素 a_t $(a_t < x)$ 容易发现，在操作范围从左往右移的过程中，当 a_t 第一次进入操作范围时，它会变成最大值，之后慢慢变小，最后又变回原来的值。因此每个数只会从 0 变成 1 一次，再从 1 变成 0 一次。
- 我们只要对每个元素找出这两次变化的位置，就能利用前缀和算出在每个位置进行操作对 1 的数量的影响。
- 总体复杂度 $\mathcal{O}(n \log A)$，其中 A 是答案可能的最大值。

B. 索道

题意

- 在距离索道入口 0 和 $(n+1)$ 单位距离的位置有索道站，给出在 $1, 2, \cdots, n$ 单位距离架设支撑塔的成本，分别是 $a_1, a_2, \cdots, a_n$。要求相邻支撑塔或索道站之间的距离必须小于等于 k。
- 成本序列会进行 q 次临时的修改（之后会复原），求出架设支撑塔的最小总成本。
- $1 \le n \le 5 \times 10^5$, $1 \le k \le \min(n+1, 3 \times 10^3)$, $1 \le q \le 3 \times 10^3$.

B. 索道

- 首先考虑没有修改的时候应该怎么做。维护 $f(i)$ 表示只考虑前 i 个位置，从位于 0 的索道站出发，把支撑塔架到了第 i 个位置的最小总成本。容易得出以下转移方程：

$$f(i) = \min_j f(j) + a_i$$

- 其中 j 需要满足以下条件：
  - $0 \le j < i$。
  - $i - j \le k$。
  - $[j+1, i-1]$ 的范围内没有必须架设的支撑塔。
- 可以把位于 $(n+1)$ 的索道站看成一个成本为 0 且必须架设的支撑塔，答案即为 $f(n+1)$，初值 $f(0) = 0$。
- 直接实现该转移方程的复杂度是 $\mathcal{O}(nk)$ 的，但我们很容易利用单调队列直接选出最优的 $\min_j f(j)$，将复杂度降至 $\mathcal{O}(n)$。

B. 索道

- 接下来考虑如何处理修改。我们发现，对位置 p 的修改，将会影响所有 $p \le i \le n+1$ 的 $f(i)$ 值。如果直接重算所有受影响的 $f(i)$ 值，总体复杂度将变为 $\mathcal{O}(nq)$。

性质

任意一段长度为 k 的连续位置中，至少有一个位置被包含在最终方案里。这个重要性质很容易证明，如果存在一段长度为 k 的连续位置都没有架设支撑塔，那么分别位于这一区间两侧的支撑塔之间的距离将大于 k。

B. 索道

- 令 $g(i)$ 表示只考虑后 $(n-i+1)$ 个位置，位置 i 已经架设了支撑塔（这个位置的成本看作 0），并一路架设到位于 $(n+1)$ 的索道站的最小总成本。如果位置 i 被包含在最终方案里，那么最终方案的总成本就是 $f(i)+g(i)$。
- 对于所有 $p \le i \le n+1$，$g(i)$ 的值不受位置 p 修改的影响，因此我们可以在所有修改开始前预处理 g 的值。因此对于位置 p 的修改，我们考虑 $[p, \min(p+k-1, n+1)]$ 这一连续区间，只重算 $p \le i \le \min(p+k-1, n+1)$ 的 $f(i)$ 值，并枚举哪个位置被包含在最终方案里。答案就是

$$\min_{i=p}^{\min(p+k-1,n+1)} f(i)+g(i)$$

。

B. 索道

- 每次重新计算依然使用单调队列优化，将 $f(p)$ 之前的不超过 k 个 $f(i)$ 值用来初始化单调队列（由于 $f(p)$ 只能从这些位置转移过来）。
- 每次修改的复杂度为 $\mathcal{O}(k)$，总体复杂度 $\mathcal{O}(n+kq)$。每次计算答案之后需要将 f 的值复原。

A. 停停，昨日请不要再重现

题意

- 给定一张 n 行 m 列的网格，在位于第 i_h 行第 j_h 列的格子上有一个洞，其它每个格子都是空地并且都有一只袋鼠。
- 所有袋鼠会同时根据按下的 U，D，L，R 按键移动，如果一只袋鼠踩到了洞或者移动到了网格外面，它将被从网格上移除。
- 给出长度为 l 的一个操作序列，若执行后网格上恰有 k 只袋鼠存留，求出有多少位置可能存在洞。
- $1 \leq n, m \leq 10^3$，$0 \leq k < n \times m$，$1 \leq l \leq 10^6$，保证所有数据 $n \times m$ 之和以及操作序列长度之和均不超过 10^6。

A. 停停，昨日请不要再重现

性质

所有袋鼠都向上移动一步，相当于上下边界向下移动一步；所有袋鼠都向左移动一步，相当于左右边界向右移动一步。

- 因此我们每一步模拟边界的移动。由于在任意一步掉出边界的袋鼠都会被移除，我们维护以下内容：
  - u 表示上边界在移动过程中最大移动到了哪一行。
  - d 表示下边界在移动过程中最小移动到了哪一行。
  - l 表示左边界在移动过程中最大移动到了哪一列。
  - r 表示右边界在移动过程中最小移动到了哪一列。
- 只有初始位置满足 $u \le i \le d$ 且 $l \le j \le r$ 的袋鼠才会被保留。因此在不考虑洞的情况下，有 $(d-u+1) \times (r-l+1)$ 只袋鼠存留。如果 $u > d$ 或 $l > r$ 则没有任何袋鼠存留。

A. 停停，昨日请不要再重现

- 利用相似的思路模拟洞的反向移动。假设洞的初始坐标是 $(0,0)$，我们把洞经过的坐标都记录下来并去重。设 (i,j) 是洞经过的坐标，由于洞真正的初始坐标是 (i_h,j_h)，因此如果 $u \le i_h + i \le d$ 以及 $l \le j_h + j \le r$，那么位于 (i_h+i, j_h+j) 的袋鼠将额外被去除。
- 因此我们枚举 i_h 和 j_h，并计算洞经过的坐标中是否恰有 $(k-(d-u+1)\times(r-l+1))$ 个坐标满足 $u-i_h \le i \le d-i_h$ 以及 $l-j_h \le j \le r-j_h$。这就是一个二维前缀和问题，在枚举坐标之前预处理前缀和即可。

A. 停停，昨日请不要再重现

- 虽然洞移动的次数至多为 $|s|$ 次，但因为洞移动的时候，边界也在移动。如果洞的行坐标的绝对值超过了 n，那么将出现 $u > d$；如果洞的列坐标的绝对值超过了 m，那么将出现 $l > r$。只要之前特判了这些情况，洞的坐标范围将受到限制，可以简单地用二维前缀和处理。
- 总体复杂度 $\mathcal{O}(|s| + nm)$。

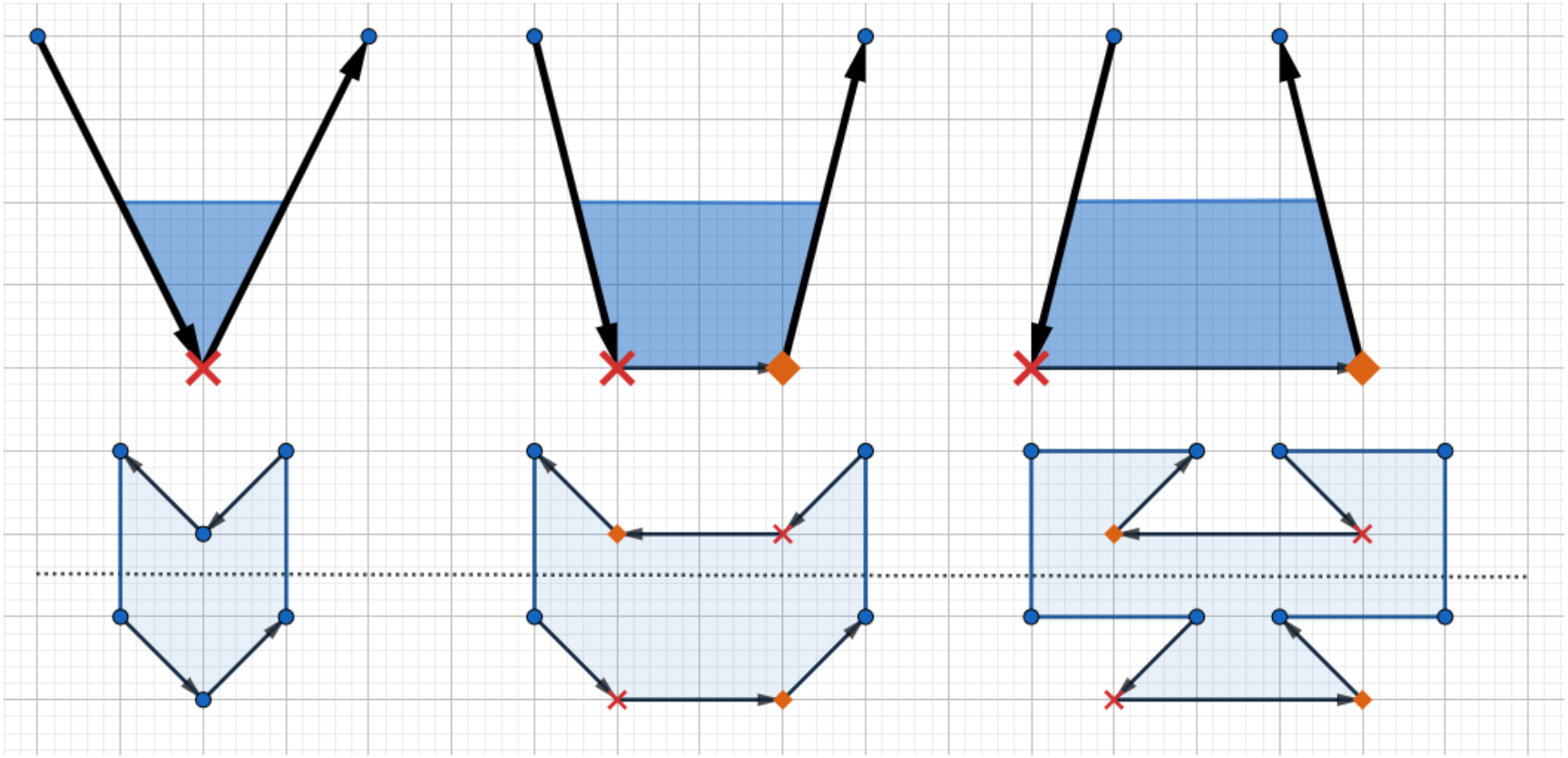
M. 清空水箱

题意

- 给定一个由 n 个顶点组成的多边形水箱，里面充满水。
- 求至少需要安装多少个出水阀门，才能在所有阀门同时打开后，让水箱里的水全部流出。
- $3 \leq n \leq 2 \times 10^3$，顶点坐标均为整数且绝对值不超过 10^4。

M. 清空水箱

- 每个局部最低点都要安装一个出水阀门，因此本题求的就是局部最低点的数量。
- 局部最低点可以按是否位于水平的平台上分成两种情况：在平台上和不在平台上。



M. 清空水箱

- 对于第一种情况，局部最低点 P 不在水平的平台上。那么
  - 以 P 为终点的向量 $\vec{v}_1$ 的 y 值应该小于 0，而以 P 为起点的向量 $\vec{v}_2$ 的 y 值应该大于 0。
  - 为了防止“天花板”上的点被错误统计，由于多边形的顶点是按逆时针顺序给出的，因此还要额外判断 $\vec{v}_1 \times \vec{v}_2 > 0$，这里 $\times$ 是向量的叉积。

M. 清空水箱

- 对于第二种情况，局部最低点 P 在水平的平台上。我们不妨以平台上的第一个点 P 为代表。这里的“第一个点”指的是以 P 为终点的向量不是水平的。
  - 以 P 为终点的向量 $\vec{v}_1$ 的 y 值应该小于 0，而从 P 开始下一条不是水平的向量 $\vec{v}_2$ 的 y 值应该大于 0。
  - 为了防止“天花板”上的点被错误统计，由于多边形的顶点是按逆时针顺序给出的，因此还要额外判断 P 的下一个点 Q。Q 的 x 坐标需要大于 P 的 x 坐标。需要注意这种情况下 $\vec{v}_1 \times \vec{v}_2$ 的符号是任意的，反而不应该判断叉积的符号。
- 枚举每个点并进行判断即可。复杂度 $\mathcal{O}(n)$。

J. 完美匹配

题意

- 给定一张包含 n 个顶点的无向图（n 是偶数）以及 n 个整数 $a_1, a_2, \cdots, a_n$。对于任意满足 $1 \le i < j \le n$ 的正整数 i 和 j，若 $|i-j| = |a_i - a_j|$（$|x|$ 表示 x 的绝对值）则在无向图中顶点 i 和顶点 j 之间连一条无向边。
- 求一个完美匹配，或表明不存在完美匹配。
- $2 \le n \le 10^5$，保证所有测试数据中 n 之和不超过 10^6。

J. 完美匹配

转换

当 $i - a_i = j - a_j$ 或 $i + a_i = j + a_j$ 时，i 和 j 有连边。我们构建这样一个二分图：每个 i 可以看做是一条边，这条边连接二分图左边编号为 $(i - a_i)$ 的点，以及右边编号为 $(i + a_i)$ 的点。此时 i 和 j 能匹配当且仅当它们在二分图里对应的边有公共顶点。

- 问题转化成将一张图分解成若干条边不相交（点可以相交）的长度为 2 的链。

J. 完美匹配

- 对于每个连通分量，如果它含有奇数条边，无解。
- 否则求出任意一棵 dfs 树，然后从深到浅考虑每一个点。找到所有和它相连的未被匹配的边，除了它连向父亲的边（这条边显然未被匹配）。如果这些边是偶数条，两两匹配即可，连向父亲的边会在处理父亲时被匹配上。如果这些边是奇数条，就把连向父亲的边也加入匹配。
- 复杂度 $\mathcal{O}(n)$。

证明

因为是自底而上贪心，构造失败当且仅当贪心匹配后，根节点依然存在一条未被匹配的边（根节点不存在连向父亲的边）。由于已经匹配的边的总数为偶数，则说明整个连通块边的总数是奇数，不符合假设。

E. 树的染色

题意

- 给一棵 n 个节点的有根树。
- 操作 i $(0 \le i \le n-1)$ 可以以 a_i 的代价，任意指定一个 u，将 u 子树里，所有的节点距离 u 距离恰为 i 的点染成黑色。
- 一开始所有 n 个节点都是白色的，需要以最小的代价将所有节点染成黑色。
- $2 \le n \le 10^5$.

E. 树的染色

观察

注意到一次染色操作只会影响某一层的节点。因此我们将每层节点分开考虑，答案就是把每层节点染黑的最小代价之和。

- 假设我们正在考虑染黑深度为 D 的节点。由于深度大的节点无法染黑深度小的节点，因此可以把深度大于 D 的节点暂时删掉，这样深度为 D 的节点就变成了树的叶子。
- 记 $f(u)$ 表示把以 u 为根的子树中，叶子全部染黑需要的最小代价。记节点 u 的深度为 d_u，记节点 u 的所有子节点形成的集合为 $\mathrm{son}(u)$，我们有如下转移方程：

$$f(u) = \min(a_{D-d_u}, \sum_{v \in \mathrm{son}(u)} f(v))$$

E. 树的染色

- 每次将所有叶子和根节点一起建立虚树。记虚树中的节点 u 在原树中的深度为 d_u，记节点 u 在虚树中的所有子节点形成的集合为 $\text{virt-son}(u)$，记节点 u 在虚树中的父节点为 p_u，则转移方程可以改写为：

$$f(u) = \min(\min_{i=d_{p_u}+1}^{d_u} a_{D-i}, \sum_{v \in \text{virt-son}(u)} f(v))$$

算法分析

- 虚树点数是 $\mathcal{O}(\text{关键点数})$，因此所有虚树的总节点数是 $\mathcal{O}(n)$。通过预处理区间最值，可以 $\mathcal{O}(1)$ 求出所需要的 $\min_{i=d_{p_u}+1}^{d_u} a_{D-i}$。由于建立虚树的总体复杂度是 $\mathcal{O}(n \log n)$ 的，因此总体复杂度为 $\mathcal{O}(n \log n)$。

C. 智巧灵蕈大竞逐

题意

- 给出一个 $n \times m$ 的矩阵，每次可以交换一对相邻元素，顺时针旋转一个 2×2 的子矩阵，或者选择一个预设对指定大小的子矩阵进行覆盖（盖章）。
- 要求构造一个操作序列，使得矩阵变成目标状态，要求操作总数不超过 4×10^5，盖章操作数量不能超过 400。
- $2 \leq n, m \leq 20$.

C. 智巧灵蕈大竞逐

- 考虑从目标状态撤销操作得到能匹配初始状态的矩阵。交换操作与旋转操作均可逆，对于盖章操作，逆操作相当于将一个能与要盖的章匹配上的子矩阵全变为通配符。

观察

由于存在交换操作，字符的位置几乎是无关紧要的，只需要对每种字符判断个数就能知道一个章是否可能盖下去。将一个字符变为通配符后，会将原来不能盖的章变得能盖（而不会倒过来使原来能盖的章变得不能盖），因此通配符越多越好。

- 可以遵循“能盖的章现在就盖”的贪心策略。如果保证每次盖章操作至少新增一个通配符，盖章次数就不会超过 nm 次，符合 400 次盖章操作的限制。

C. 智巧灵蕈大竞逐

- 不妨选定矩阵左上角进行盖章。通过交换操作使得左上角矩阵与要盖的章，使用操作将其全变成通配符。接下来每次移动一个章里有的字符到左上角矩阵对应位置，再进行盖章操作就能新增一个通配符。一直重复直到无法新增通配符。这个章之后也都用不上了。接下来找到下一个能盖的章，重复以上操作。
- 当所有的章都已经用不上时，检查当前矩阵每种字符的数量判断是否可能经过若干次交换操作后与初始矩阵进行匹配。如果可能匹配则有解，并构造交换方案，否则无解。

C. 智巧灵蕈大竞逐

- 如果把初始矩阵也视为一个章，则“盖一个新章”操作将进行 $(k+1)$ 次，每次可能会移动矩阵中所有的字符（也就是 nm 个字符），而每个字符移动到目标位置最多会花 $(n+m-1)$ 步。因此，“盖一个新章”最多消耗 $nm(n+m-1)(k+1)$ 步交换。
- 而“用旧章转化一个字符”最多进行 nm 次，每次只移动给一个字符到目标位置，最多花费 $(n+m-1)$ 步。因此，“用旧章转化一个字符”最多消耗 $nm(n+m-1)$ 步交换。
- 因此，所有操作最多总共消耗 $nm(n+m-1)(k+2)$ 步交换，加上 400 次盖章操作也符合 4×10^5 次操作的限制。

C. 智巧灵蕈大竞逐

- 在模拟过程中，需要用一个数组动态维护当前矩阵中每种字符的个数，以及通配符的数量。
- 由于每次操作都至少将一个字符变为通配符，我们需要检查 nm 次当前章是否还能盖，复杂度是 $\mathcal{O}(nm|\Sigma|)$，$|\Sigma|$ 表示字符集的大小。
- 换章将会进行 $(k+1)$ 次，直接枚举复杂度是 $\mathcal{O}(k^2|\Sigma|)$。
- “盖一个新章”操作将进行 $(k+1)$ 次，每次暴力移动所有目标字符，复杂度是 $\mathcal{O}(n^2m^2(n+m))$。
- “用旧章转化一个字符”操作将进行 nm 次，每次暴力移动目标字符，复杂度是 $\mathcal{O}(n^2m^2(n+m))$。
- 因此总体复杂度为 $\mathcal{O}(n^2m^2(n+m)+(nm+k^2)|\Sigma|)$，大概不到 10^7，能很快运行完成。

H. 工厂重现

题意

- 给出一棵节点数量为 n 的树，边为无向边，有边权。
- 要求选出大小为 k 的点集 S，最大化 $\sum_{i\in S}\sum_{j\in S} dis(i,j)$。$dis(i,j)$ 表示节点 i 与节点 j 的距离。
- $2 \le n \le 10^5$，$2 \le k \le n$.

H. 工厂重现

- 任选一个点定根，对于一条连接 u 和 fa_u 的权值为 w_u 的边，如果 u 的子树内选了 $i\ (0 \le i \le k)$ 个关键点，那么这条边对关键点两两距离之和的贡献是 $w_u i(k-i)$。
- 记 $dp_{u,i}$ 表示以 u 为根的子树里选了 $i\ (0 \le i \le k)$ 个关键点时子树里每条边贡献之和的最大值，现在把一棵以 v 为根且包含 $j\ (0 \le j, i+j \le k)$ 个关键点的子树合并上来，可以得到转移方程

$$tdp_{u,ti} = \max_{i+j=ti}\{dp_{u,i} + dp_{v,j} + w_v j(k-j)\}$$

- 这里 tdp_u 即为以 v 为根的子树合并到 u 上之后的 dp_u。在没有子树合并到 u 上时，只需要考虑是否选 u，因此初始有 $dp_{u,0} = dp_{u,1} = 0$。

H. 工厂重现

- 由于两个上凸数组对应位置相加的结果仍然是上凸数组，两个上凸数组的 $(\max, +)$ 卷积的结果仍然是上凸数组（实际就是 *Minkowski Sum*），可以归纳证明所有 dp_u 都是上凸的，也就是差分数组总是单调不增的，可以使用启发式合并求出每个 dp_u 的差分数组。
- 由于需要支持对一个子树里的差分数组打上加等差数列的标记，需要手写平衡树实现加等差数列的标记下传，使用 *Splay Tree* 进行启发式合并可以做到 $O(n \log n)$ 的复杂度。

K. 堆里的 NaN

题意

- 考虑从 1 到 $(n-1)$ 的所有整数再加上一个额外的元素 NaN。称这 n 个元素的一个排列 P 是“堆序列”，若存在这 n 个元素的一个排列 Q 满足 $P = \texttt{HEAPIFY}(Q)$。$\texttt{HEAPIFY}(Q)$ 为将 Q 的元素依次插入小根堆 H 后，H 的序列表示。
- 现在从这 n 个元素的所有排列中等概率随机选取一个，求被选中的排列是堆序列的概率。
- 不超过 $1 \le T \le 10^3$ 组测试数据，每组测试数据满足 $1 \le n \le 10^9$。

K. 堆里的 NaN

结论

给定一棵 n 个节点的树，在每个节点中填一个从 1 到 n（含两端）的数，要求每个节点填的数都不一样，且后代节点的数要比祖先节点的数大。方案数为 $\frac{n!}{\prod\limits_{u\in\mathbb{T}} s_u}$，其中 $\mathbb{T}$ 表示树的所有节点构成的集合，s_u 表示以 u 为根的子树中有几个节点。该式子即 n 的阶乘除以每个子树大小的乘积。

转化

给定一棵 n 个节点的完全二叉树，选择一个节点 u，将完全二叉树分为以 u 为根的子树 U 和子树之外的部分 U'。将 0 填入节点 u 中，还需要将 1 到 $(n-1)$ 填入 U 和 U' 中，使得 U 和 U' 分别满足后代节点的数要比祖先节点的数大。求方案数。

K. 堆里的 NaN

- 尝试套用上面的常用结论计算答案。设 $\binom{n}{m}$ 表示从 n 个物品里选 m 个的组合数，P 表示 U 中每个子树大小的乘积，P' 表示 U' 中每个子树大小的乘积，答案为

$$\frac{s_u!}{P} \times \frac{(n-s_u)!}{P'} \times \binom{n-1}{s_u-1} = \frac{(n-1)! \times s_u}{P \times P'}$$

- 不同 0 的位置对应不同的方案，则总方案数就是 $\sum_{u=1}^{n} \frac{(n-1)! \times s_u}{P \times P'}$，因此概率为 $\sum_{u=1}^{n} \frac{s_u}{P \times P' \times n}$（除以 $n!$）。

K. 堆里的 NaN

- 直接计算这个式子的复杂度至少是 $\mathcal{O}(n)$ 的，但完全二叉树中有许多同构的子树，可以帮助我们减少计算。
- 称完全二叉树中编号最大的点以及它的祖先节点为“特殊点”，除了特殊点以外，以其它任意一个点为根的子树都是满二叉树。
- 另外还可以发现，除了节点 1 以外，每一个特殊点都有一个根节点与它同深度的满二叉子树。称该满二叉树为该特殊点的“兄弟树”。

K. 堆里的 NaN

- 因为兄弟树是满二叉树，所以我们只关心它的高度。设整棵树高度为 D，某个特殊点的深度为 d。如果该特殊点是父节点的左子节点，那么兄弟树的高度为 $(D-d)$；如果该特殊点是父节点的右子节点，那么兄弟树的高度为 $(D-d+1)$。
- 令 $sh(d)$ 表示深度为 d 的特殊点的兄弟树的高度，$sz(d)$ 表示以深度为 d 的特殊点为根的子树中有几个节点。
- 通过递推预处理以下值：
  - $g(i)$ 表示一棵高度为 i 的满二叉树，所有子树大小的乘积。
  - $h(i,j)$ 表示一棵高度为 i 满二叉树，再去掉任意一棵高度为 j 的子树（要求 $j \leq i$。也就是说，现在这棵树有 (2^i-2^j) 个节点），所有子树大小的乘积。

K. 堆里的 NaN

- 整棵二叉树子树大小的乘积为

$$X = (\prod_{d=1}^{D} sz(d)) \times (\prod_{d=2}^{D} g(d))$$

- 预处理的复杂度是 $\mathcal{O}(\log^2 n)$。

K. 堆里的 NaN

- 接下来我们讨论两种情况：u 是特殊点，以及 u 不是特殊点。
- 如果 u 是特殊点，设该特殊点深度为 d，则所有以深度小于 d 的特殊点为根的子树的大小都将减少 $sz(d)$，而其它子树的大小不受影响。也就是说

$$s_u = sz(d)$$

$$P \times P' = X \div \left(\prod_{i=1}^{d-1} sz(i)\right) \times \left(\prod_{i=1}^{d-1} (sz(i) - sz(d))\right)$$

- 特殊点只有 $\log n$ 个，枚举所有特殊点即可。这一部分的复杂度是 $\mathcal{O}(\log^2 n)$。

K. 堆里的 NaN

- 如果 u 不是特殊点，那么它一定位于某个特殊点的兄弟树中。设它位于深度为 d 的特殊点的兄弟树中，且 u 的子树高度为 w，则所有以深度小于 d 的特殊点为根的子树的大小都将减少 $(2^w - 1)$，当然该兄弟树内部的节点也会受到影响，而其它子树的大小不受影响。
- 为了处理这一情况，我们另外计算

$$f(d, w) = \prod_{i=1}^{d} (sz(d) - 2^w + 1)$$

- 那么

$$s_u = 2^w - 1$$

$$P \times P' = X \div (\prod_{i=1}^{d-1} sz(i)) \times f(d-1, w) \div g(sh(d)) \times h(sh(d), w)$$

K. 堆里的 NaN

- 该式子可以 $\mathcal{O}(1)$ 计算。我们枚举特殊点以及 w 即可。另外，满足 d 和 w 的点 u 共有 $2^{sh(d)-w}$ 种，所以这一情况求出来的式子要乘以该值。这一部分的复杂度也是 $\mathcal{O}(\log^2 n)$。
- 答案就是两种情况加起来，另外除法需要通过逆元计算。总体复杂度 $\mathcal{O}(\log^2 n \log M)$。

L. 命题作文

题意

- 有一张由 n 个点与 $(n-1)$ 条边构成的无向连通图 G。顶点编号从 1 到 n（含两端），第 i 条边连接顶点 i 与 $(i+1)$。
- 接下来向图中依次加入额外 m 条边。每次加入一条额外边后，求从图中选择两条边 e 与 f 的方案数，要求满足如果边 e 与 f 同时被删除，图将变得不连通。
- 保证所有数据中 n 之和与 m 之和均不超过 2.5×10^5。

L. 命题作文

- 注意到额外边 (u, v) $(u \le v)$ 与链边 $(u, u+1) \ldots (v-1, v)$ 构成一个环。我们称这条额外边覆盖这些链边。
- 一对满足条件的边 (e, f) 符合如下条件之一：
  - e 与 f 中有至少一条割边（是链边且没有被任何额外边覆盖）。
  - e 与 f 中有一条额外边，另一条边是条链边，且该链边只被这条额外边覆盖。
  - e 与 f 都是链边，且所有额外边要么同时覆盖 e 与 f，要么同时不覆盖 e 与 f（被覆盖情况完全相同）。
- 前两类可以用并查集维护。接下来我们考虑如何维护第三类情况。

L. 命题作文

- 我们用双向链表把所有被覆盖情况完全相同的链边串在一起。那么每次加入一条额外边时，本质是有些双向链表需要被切开。
- 注意到被切开的位置始终是某个边本身被覆盖，且和它的前驱（或后继）没有被覆盖的情况。我们用线段树维护区间最远前驱和最远后继，就能 $\mathcal{O}(cnt \log n)$ 地找到所有被切开的位置，其中 cnt 是被切的位置数量。由于每次被切会增加双向链表的数量，且最多只有 $n-1$ 条链（每个链边单独一条），因此这里复杂度为 $\mathcal{O}(n \log n)$ 。
- 对于每条被切的双向链表，我们需要维护它的大小。这里用启发式分裂即可，复杂度为 $\mathcal{O}(n \log n)$ 。
- 注意双向链表有中间某段被切出来的情况，此时两个切点都会被上述线段树找到。

F. 三角形

题意

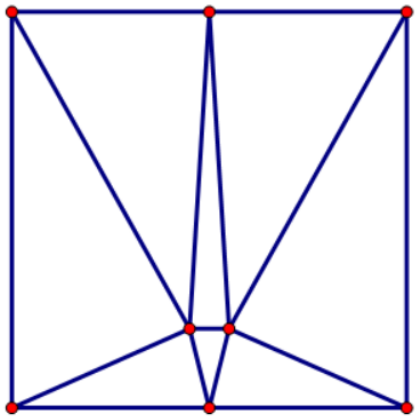
- 给定一个正整数 k，将一个正方形分割成恰好 k 个锐角三角形。
- $1 \leq k \leq 50$.

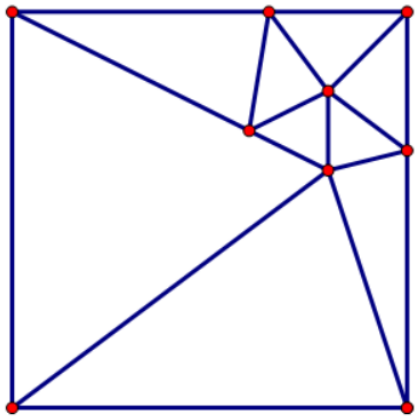
F. 三角形

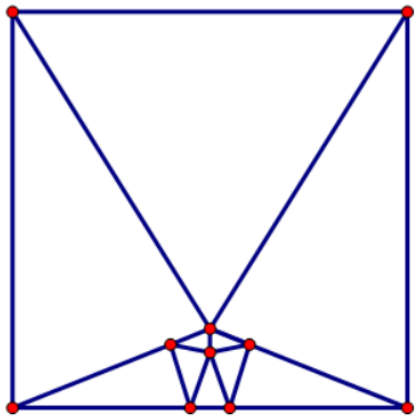
- 可以证明至少需要 8 个锐角三角形才能构成一个单位正方形。
- 如果我们可以将单位正方形划分成 k 个锐角三角形，那么我们可以选择任意一个锐角三角形并将其三条边上的中点两两相连，从而得到一个将正方形划分成 $k+3$ 个锐角三角形的方案。

F. 三角形

- 下面给出 $k = 8, 9, 10$ 的其中一种构造。







Thank you!