2023 国际大学生程序设计竞赛亚洲区域赛（济南站）

SUA 程序设计竞赛命题组

2023 年 12 月 3 日

概况

- 阶段一（Easy）：D、I、A
- 阶段二（Medium-Easy）：G、K、M
- 阶段三（Medium）：E、B
- 阶段四（Medium-Hard）：H、C、L、F
- 阶段五（Hard）：J

D. Largest Digit

题意

- 令 $f(x)$ 为 x 十进制表示下的最大数码，求 $\max f(a+b)$，其中 $a \in [l_a, r_a]$，$b \in [l_b, r_b]$。

- 如果 $r_a - l_a + 1 \geq 10$ 或 $r_b - l_b + 1 \geq 10$，则 $(a+b)$ 的个位数会从 0 到 9 都出现一遍，因此答案就是 9。剩下的情况直接暴力枚举即可。
- 复杂度 $\mathcal{O}(B^2 \log X)$，其中 $B = 10$，$X = 10^9$。

I. Strange Sorting

题意

- 给定排列 $a_1, a_2, \cdots, a_n$ 表示一个 n，您需要将该排列按升序排序。
- 为此可以执行以下操作至多 $\lfloor \frac{n}{2} \rfloor$ 次：选择两个下标 l 和 r 满足 $1 \le l < r \le n$ 以及 $a_l > a_r$，将 $a_l, a_{l+1}, \cdots, a_r$ 按升序进行排序。

I. Strange Sorting

- 首先，对于形如 $2, 1, 4, 3, 6, 5, \ldots$ 的序列，任意操作只能在相邻的逆序对之间进行，因此 $\lfloor \frac{n}{2} \rfloor$ 次操作是必须的。现在我们用以下的方式操作：
- 找到最小的 u，满足 $u \neq a_u$（若没有这样的 u，则序列已经有序），然后找到最大的 v，满足 $a_v < a_u$，排序区间 $[u, v]$。接下来我们证明，这样的操作最多只需要 $\lfloor \frac{n}{2} \rfloor$ 次循环就能排序原序列。
- 我们证明，在做了一次 $[u, v]$ 操作后，可以使 $a_u = u$ 且 $a_{u+1} = u + 1$，这样能使排好序的位置至少增加两个：如果 $a_u = u + 1$，选择 $a_v = u$ 即可，显然我们选择最大的 v 不会更劣，这里的优劣指的是满足 $a_i = i$ 的最长前缀的长度；如果 $a_u \geq u + 2$，选择 u 和 $u + 1$ 这两个元素所处位置的较大者即可，同样，我们选择最大合法 v 的操作方式不会劣于这种操作。
- 因此，这样只需要最多 $\lfloor \frac{n}{2} \rfloor$ 次循环即可排序完成（奇数情况下，最后一个剩余的元素是自然满足条件的）。

A. Many Many Heads

题意

- 给定一个括号序列中每个位置的种类（方/圆）
- 问是否存在唯一的方法定向括号序列，使得其为一个合法的括号序列。

A. Many Many Heads

- 注意到如下判定一个括号序列是否合法的方法：维护一个栈，按照顺序遍历序列中每一个括号，如果此时栈非空且栈顶的括号种类等于该括号，则弹出栈顶，否则将该种类括号压入栈。
- 如果原括号序列可以通过定向得到合法的括号序列，那么栈一定非空。
- 通过考察该方法，可以发现，如果在原序列中存在三个连续的位置是同一种括号，那么方案数一定不唯一。

A. Many Many Heads

Proof

注意到对于三个连续的种类相同的位置 abc（不妨假设都为圆括号）上述方法得到的合法括号序列在这三个位置一定形如 ()(或)()
对于前者，找到 c 的左括号所匹配的右括号的位置，假设形如 $()(X)$，那么一定可以调整成 $(()X)$。
对于后者，找到 a 的右括号所匹配的做括号的位置，假设形如 $(X)()$，那么一定可以调整成 $(X())$。
因此此时必定有另一种不同的定向方法。

A. Many Many Heads

- 因此，如果我们写下来每个位置的种类，那么不能有长度大于等于 3 的连续段。
- 接下来考察长度等于 2 的连续段。如果只有一段，那么显然是合法的。
- 如果有两段，假设形如 $A()B()C$，那么可以证明，唯一合法的括号序列一定形如 $[()]()$。
- 这是因为：
  - 如果 $|B| = 1$，那么显然其只能是方括号（不妨假设为右方括号，左方括号同理），那么 A 形如 $A'[$，整个串形如 $A'[()]()C$。
  - 若 A' 非空，则 A' 只能以圆括号结尾。容易发现无论这个括号如何匹配，其均可以与 C 前两个括号作调整。因此 A' 必定为空。同理 C 也必须为空，序列只能形如 $[()]()$。
  - 若 $|B| > 1$，容易同样通过类似的调整来证明一定存在至少一种方案

A. Many Many Heads

- 对于有大于等于三段的情况，同样可以类似的调整来证明一定不合法。
- 综上所述，一个括号序列合法，当且仅当其不存在长度大于等于 3 的连续段，且存在不超过 2 个长度大于等于 2 的连续段。
- 直接进行判定即可，时间复杂度为 $O(|S|)$。

G. Gifts from Knowledge

题意

- 给一个 $n \times m$ 的 01 矩阵，可以选择反转一些行（第一列变成最后一列，以此类推），目标是每一列最多有一个 1。求方案数。

- 如果第 j 列和第 $(m-j+1)$ 总共的 1 的数量超过了两个，显然无解。否则假设这两个 1 位于第 i 行和第 i' 行，有两种情况：
  - 两个 1 处于同一列，则 i 和 i' 的选择情况是相反的。
  - 两个 1 处于不同列，则 i 和 i' 的选择情况是相同的。
- 给定相同相反关系，求方案数是一个经典问题，可以通过建图来维护。

- 考虑一张 $2n$ 个点的图，其中点 $1 \le i \le n$ 表示选了第 i 行，点 $(i+n)$ 表示不选第 i 行。
- 如果 i 和 i' 相同，则连边 (i, i') 和 $(i+n, i'+n)$；如果 i 和 i' 不同，则连边 $(i, i'+n)$ 和 $(i', i+n)$。
- 如果最后 i 和 $(i+n)$ 处于同一连通块则无解，否则设连通块数量为 $2c$，则答案为 2^c。

K. Rainbow Subarray

题意

- 称连续子数组 $a_l, a_{l+1}, \cdots, a_r$ 是彩虹子数组，若 $a_{i+1} - a_i = 1$。可以进行至多 k 次操作，每次操作可以把一个元素增加或减少 1，求操作后最长的彩虹子数组。

- $a_{i+1} - a_i = 1$，等价于 $a_i - i = a_j - j$，因此先把每个元素减去它们的下标，问题变为操作后最长的相等子数组。
- 把一个数组里所有元素都变得相同至少需要几次操作？这是一个经典问题，把所有数都变成中位数即可（如果长度为偶数，则中间两个数任选一个）。
- 如果把一个数组变成相等的至多需要 k 次操作，那么它的子数组也至多只要 k 次操作即可变成相等的。因此本题可以用双指针（滑动窗口）求解。滑动窗口的中位数用两个 set 求中位数的技巧维护即可。复杂度 $\mathcal{O}(n \log n)$。

M. Almost Convex Polygon

题意

- 给定点集合 S，其中任意两点的坐标都不相同，且任意三点不共线。
- 称一个多边形 P 为近似凸多边形，当且仅当多边形 P 是简单多边形，且多边形的顶点属于 S，且 S 中所有点要么在多边形内部，要么在多边形的边界上。
- 令 $\mathbb{U}$ 表示所有近似凸多边形构成的集合，多边形 R 使得 $|R|$ 在 $\mathbb{U}$ 的所有多边形中是最小的，计算多边形 $Q \in \mathbb{U}$ 的数量，满足 $|Q| \le |R| + 1$。

M. Almost Convex Polygon

- 首先求出 S 的凸包作为 P。
- 首先 P 本身就是可能的 Q，然后要统计所有满足 $|Q| = |P| + 1$ 的 Q。
- 不难发现 Q 仍然要包含 P 的所有点，只能在 S 中再选一个不是 P 的顶点的点作为 Q 的顶点。
- 枚举新加的点 w，只能从 P 上选两个相邻的点 u 和 v，切断 $u - v$ 然后连接 $u - w$ 和 $v - w$。
- 这要求 w, u, v 构成的三角形区域内没有其他 S 内的点。
- 问题转化为找出所有合法的 (w, u, v)。

M. Almost Convex Polygon

- 这里有两种做法：
  - 第一种做法是枚举 S 中不是 P 的顶点的点 w 之后将其他所有点以 w 为极点进行极角排序，当 P 上相邻两个点 u 和 v 具有相邻的极角序时，这组 (u, v, w) 是合法的。
  - 第二种做法是枚举 P 上相邻两个点 u 和 v，对每个 S 中所有不是 P 的顶点的点 w，分别计算 $w - u - v$ 和 $w - v - u$ 的夹角，那么当 w, u, v 构成的三角形之间的包含关系等价于两个夹角的二维偏序，使用单调栈可以求出所有合法的 (w, u, v)。
- 以上两种做法的复杂度均为 $O(n^2 \log n)$。

E. I Just Want... One More...

题意

- 给定一张平衡二分图。您需要添加恰好一条边，连接 U 中的一个节点与 V 中的一个节点，使得图的匹配数增加。求方案数。
- $1 \leq n, m \leq 10^5$，$1 \leq \sum(n+m) \leq 4 \times 10^5$

E. I Just Want... One More...

- 建源 S 向二分图左侧每个点连流量为 1 的边，汇 T 从二分图右侧每个点连流量为 1 的边，原图的匹配数就是新图的最大流。
- 先求出新图最大流以及对应的残余网络，添加一条边能够提高匹配数当且仅当新的残余网络上源 S 和汇 T 联通。因此求出源 S 能到达左侧多少点，右侧多少点能到达汇 T，相乘即为答案。

B. Graph Partitioning 2

题意

- 给一颗 n 个点的树，还有一个整数 k
- 问有多少种方式删除这棵树的一些边
- 使得剩下的每个连通块的大小都是 k 或 $k+1$.

B. Graph Partitioning 2

- 若 $k \le \sqrt{n}$，记录当前包含子树根的还没切出去的块大小做树上背包，复杂度是 $\mathcal{O}(n\sqrt{n})$；
- 若 $k > \sqrt{n}$，仍然考虑做树上背包，但是记录当前子树里切出去大小为 k 的块有 i 个，并且有 $j(=0,1)$ 个包含子树根的大小为 $k+1$ 的块还没切出去，显然 $i < \sqrt{n}$，另外此时也能算出已经切出去大小为 $(k+1)$ 的块有 $\lfloor \frac{\text{size}-ik}{k+1} \rfloor - j$ 个，复杂度也是 $\mathcal{O}(n\sqrt{n})$，最坏的情况出现在树是一条链的时候。
- 实际上可以合并上述两种情况的分析，直接记录当前包含子树根的答案不为 0 的块大小进行树上背包，复杂度也是 $\mathcal{O}(n\sqrt{n})$，但是直接使用 `unordered_map` 可能会被卡常数。
- 一个可行的办法是在 k 比较小时直接树上背包，k 比较大时使用 `unordered_map` 记录答案不为 0 的值进行树上背包。

H. Basic Substring Structure

题意

- 令 $\mathrm{suf}(s, i)$ 表示 s 从第 i 个字符开始的后缀。令 $g(i) = \mathrm{lcp}(s, \mathrm{suf}(s, i))$。
- 对每个下标，求如果必须改掉第 i 个字符，$\sum g(i)$ 的值最大是多少。

H. Basic Substring Structure

- 本题的重点在于分类讨论。考虑修改第 t 个字符会对哪些 g 产生改动。
  - 若 t 在其中一个字符串内部，即 $1 \le t \le g(i)$ 或 $i \le t < i + g(i)$，则 $g(i)$ 会变成 t。
  - 否则，若 t 恰好是其中一个字符串末尾，比如 $t = g(i) + 1$，那么第 t 个字符只能变成 $s_{i+g(i)}$ 才能让 $g(i)$ 的值增加，变成其它字符则 $g(i)$ 不变。新 g 值的计算可以用后缀数组、哈希 + 二分等方式实现。$t = i + g(i)$ 的情况同理。
- 因此维护每个位置变成每个字符对答案产生的贡献（情况一需要离线维护区间加等差数列），最后离线把答案算出来即可。
- 注意由于字符集很大，实现中不能真的枚举每个字符，而是只记录对答案有影响的字符。
- 复杂度 $\mathcal{O}(n \log n)$。

题意

- 给定 m, d，构造一张 m 条边的简单无向图，满足每个点的度数不超过 d，且最大化极大独立集的数量。
- $2 \leq n \leq 20, 2 \leq d \leq m$。

C. Turn on the Light 2

- 假设我们构造出了图 G。
- 任取 G 中一棵生成树 T，对于每条非树边 $u-v$，新增一个点 v'，删去 $u-v$ 这条边之后连上 $u-v'$。
- 这样可以得到一个新图 G'。观察到 G' 中非新增点的度数与 G 中对应点度数相同，新增点度数都是 1 并且一定和非新增点相连，因此 G' 仍然满足每个点的度数不超过 d 的限制。
- 对于一个 G 的极大独立集，G' 中非新增点与 G 中对应点保持相同的选取状态，新增点的状态指定为相邻点的状态取反，就得到了一个 G 的所有极大独立集到 G' 的所有极大独立集的单射。
- 也就是说 G' 的极大独立集个数不少于 G，于是只需要考虑 m 条边也就是 $m+1$ 个点的树。

- 由于不超过 21 个点的不同构无根树的数量不到 4×10^6，直接枚举出所有树是可以接受的。
- 先考虑枚举有根树，只需要枚举根的各个子树大小的划分之后再根据子树大小选取子树即可，这里多个大小相同的子树要按顺序依次选取，避免按照不同顺序选出同一种组合。
- 对于一棵不超过 21 个点的无根树，以重心定根之后每个子树是不超过 10 个点的有根树，在枚举出所有不超过 10 个点的有根树之后，仍然按照类似的方法枚举子树的组合即可。

C. Turn on the Light 2

- 在枚举出一棵树之后还需要计算这棵树的极大独立集个数。
- 任意定根之后考虑 $dp_{u,i}$ 表示以 u 为根子树里 u 的状态是 i 的极大独立集个数即可，其中各状态含义如下：
  - $i = 0$ 表示 u 被选取；
  - $i = 1$ 表示 u 不选并且不存在 u 的儿子被选取；
  - $i = 2$ 表示 u 不选但是存在 u 的儿子被选取。
- 容易发现本质不同的输入比较少，实现不够高效的情况下可以在本地打表后提交。

题意

- 有 $(n+1)$ 个点排成一条线，编号从 0 到 n。还有 n 条线段，第 i 条线段连接点 $(i-1)$ 和 i。
- 给定 q 个区间 $[l_i, r_i]$，每个区间还有一个分数 v_i。可以选择将一些线段涂红，如果点 l_i 到 r_i 之间的线段都是红的，那么得 v_i 分。求恰好涂红 $1, 2, \cdots, n$ 条线段的最大得分。

- 设 $f(i,j)$ 表示前 i 条线段涂了 j 条，而且第 i 条线段是红色的最大得分；$g(i,j)$ 表示前 i 条线段涂了 j 条，而且第 i 条线段没涂的最大得分。有转移方程：

$$f(i,j) = \max(g(i',j-i+i') + w(i',i))$$
$$g(i,j) = \max(f(i-1,j), g(i-1,j))$$

- 其中 $i' < i$，$w(i',i)$ 表示把第 $(i'+1)$ 到第 i 条线段都涂红能得几分。朴素 dp 的时间复杂度 $\mathcal{O}(n^3+nm)$。考虑如何优化。

L. Ticket to Ride

- 设 $j' = j - i + i'$，得 $i - j = i' - j'$，也就是说只有 $(i - j)$ 相同的 g 会影响 f。考虑通过某种数据结构直接选出最优的 g。
- 考虑我们从小到大枚举 i 的时候，$w(i', i)$ 的值怎么变化。如果点 i 恰好是一个区间的终点，设该区间的起点为 l，则所有 $i' \leq l$ 的 $w(i', i)$ 都会增加 v。
- 也就是说这个数据结构需要支持三种操作：往末尾添加一个数，前缀加某个正数，求所有元素最大值。如果用线段树，时间复杂度 $\mathcal{O}(n^2 \log n + nm)$，考虑如何继续优化。

L. Ticket to Ride

- 设 m_i 表示数据结构中前 i 个数中最大值的下标，容易发现数组 M 是一段一段的，每一段都是相同的下标，而且每一段的下标对应的那个值是递增的。
- 因为每次加的都是正数，我们可以维护每一段的下标对应的值，距离前一段对应的值还有多少差距。每次前 i 个数 $+v$ 的操作，实际上就是 i 下一段的差距减少了 v。如果差距减少到 0 就和下一段合并。而且每次加的都是正数，因此只有合并段，没有分裂段。用并查集维护每个数属于哪一段即可。
- 这样就把时间复杂度优化到了 $\mathcal{O}(n(n+m)\alpha(n))$，其中 $\alpha(n)$ 是并查集复杂度中的反阿克曼函数。
- 由于本题的特殊空间限制，无法直接开出空间为 $\mathcal{O}(n^2)$ 的数组。dp 的时候第一维需要先枚举 $(i-j)$ 的值，这样就能用滚动数组计算 dp，空间复杂度降至 $\mathcal{O}(n+m)$，这也是为了让缓存更友好以提高实际运行效率。

F. Say Hello to the Future

题意

- 有一个长度为 n 的序列 a，你需要将它划分为若干个区间，满足对于每个 i，它所在的区间长度 $\geq a_i$。
- 设 $f(a)$ 表示满足上述条件的情况下划分出区间的方案数。
- 对每个 $x = 1, 2, \ldots, n$，设 b 为 a 中将 a_x 改为 1 后得到的序列，你需要求出 $f(b)$ 取模 $998\,244\,353$。
- $1 \leq n \leq 2 \times 10^5$。

F. Say Hello to the Future

- 考虑不修改序列的情况怎么做。
- 相当于每个区间 $[l, r]$ 要满足 $r - l + 1 \geq \max\{a_{l \dots r}\}$。
- 设 dp_i 表示 $a_{1 \dots i}$ 被划分为若干个区间的数量。
- 我们需要对于所有合法的区间 $[j, i]$ 进行转移，但直接使用数据结构难以维护。

F. Say Hello to the Future

- 考虑分治，设当前区间为 $[l, r]$，我们需要处理所有跨过 mid 的转移。
- 设 $mxL_i = \max\{a_{i\ldots mid}\}$，$mxR_i = \max\{a_{mid+1\ldots i}\}$。
- 那么一个跨过 $[l, r]$ 的区间合法当且仅当 $r - l + 1 \geq \max\{mxL_l, mxR_r\}$。
- 即 $r \geq l + mxL_l - 1, l \leq r - mxR_r + 1$。
- 这是一个二维偏序的形式，使用树状数组维护即可做到 $O(n \log n)$。
- 总时间复杂度为 $O(n \log^2 n)$。

F. Say Hello to the Future

- 再考虑一般的情况怎么做。
- 继续利用上述分治结构，先求出每个前后缀的答案 f, g。
- 如果我们将 a_x 改为 1，我们需要计算所有包含 x 的合法区间 $[l, r]$ 对应的 $f_{l-1} + g_{r+1} + 1$ 的最大值。
- 因此只需要考虑分治树上所有包含 x 的区间来计算贡献。

F. Say Hello to the Future

- 不妨设 $x \in [l, mid]$，那么我们需要修改 mxL 中的若干个值，并重新计算这些被修改的值带来的贡献。
- 这依然是一个二维偏序问题，每个被修改的值会产生 $O(\log n)$ 的代价。
- 对于一个分治树上的区间 $[l, r]$，依然考虑它的左半部分 $[l, mid]$。可以发现一个 mxL_i 最多只在一个 x 处被修改，因此 mxL 变化量的总和是 $O(mid - l)$ 的。
- 因此直接二维偏序计算答案即可，时间复杂度 $O(n \log^2 n)$。

J. Computational Intelligence

题意

- 给定二维平面上的两条直线段，分别从两条线段上等概率选取一个点，计算这两个点距离的期望。
- 坐标绝对值不超过 $1\,000$，要求相对误差不超过 10^{-9}，单个测试数据文件里至多有 10^5 组测试数据。

- 方便起见以下不区分点和向量的表示。
- 记两条线段分别为 P_0Q_0 和 P_1Q_1，要计算

$$\int_0^1 \int_0^1 |((1-x)P_0 + xQ_0) - ((1-y)P_1 + yQ_1)| \mathrm{d}x\mathrm{d}y$$

- 其中 $|XY|$ 表示向量 $Y-X$ 的模长，对于二维平面则是两点之间的距离。

- 考虑积分 $\int_0^\pi |\cos(\theta-\gamma)|\mathrm{d}\theta=2$。
- 记 $R_0(x)=(1-x)P_0+xQ_0$，$R_1(y)=(1-y)P_1+yQ_1$，于是有

$$
\begin{aligned}
&\int_0^1\int_0^1 |R_0-R_1|\mathrm{d}x\mathrm{d}y\\
=&\int_0^1\int_0^1\frac{1}{2}\int_0^\pi |\cos(\theta-\gamma_{R_0R_1})||R_0-R_1|\mathrm{d}\theta\mathrm{d}x\mathrm{d}y\\
=&\frac{1}{2}\int_0^\pi\int_0^1\int_0^1 |(R_0-R_1)\cos(\theta-\gamma_{R_0R_1})|\mathrm{d}x\mathrm{d}y\mathrm{d}\theta
\end{aligned}
$$

- 其中 $\gamma_{R_0R_1}$ 表示向量 R_0R_1 的极角，亦即从 x 轴正方向逆时针旋转到 R_0R_1 方向的有向角。

J. Computational Intelligence

- 记 $f(\theta) = \int_0^1 \int_0^1 |(R_0 - R_1)\cos(\theta - \alpha_{R_0R_1})|\mathrm{d}x\mathrm{d}y$
- 其几何意义为将积分区域内每个微元对应的线段 R_0R_1 投影到向量 $U_\theta = (\cos(\theta), \sin(\theta))$ 上，于是有

$$f(\theta) = \int_0^1 \int_0^1 |((1-x)P_0 \cdot U_\theta + xQ_0 \cdot U_\theta) - ((1-y)P_1 \cdot U_\theta + yQ_1 \cdot U_\theta)|\mathrm{d}x\mathrm{d}y$$

- 这里 $X \cdot Y$ 表示向量 X 和向量 Y 的点积，其结果是一个标量，于是得到关于投影角度 θ 的一维问题。
- 记 $p_0(\theta) = P_0 \cdot U_\theta$，$q_0(\theta) = Q_0 \cdot U_\theta$，$p_1(\theta) = P_1 \cdot U_\theta$，$q_1(\theta) = Q_1 \cdot U_\theta$，于是有

$$f(\theta) = \int_0^1 \int_0^1 |((1-x)p_0 + xq_0) - ((1-y)p_1 + yq_1)|\mathrm{d}x\mathrm{d}y$$

J. Computational Intelligence

- 考虑拆开绝对值，这需要对 x 和 y 上的积分区间讨论相离、相交和包含三种相对位置关系。
- 如果事先在 P_0, Q_0, P_1, Q_1 这四个点中两两枚举不同的点计算极角，就可以将 $[0, \pi)$ 分成若干个角度区间，θ 在每个角度区间内变化时 x 和 y 上的积分区间的相对位置关系不会发生变化。
- 如果能对每种情况求出 $f(\theta)$ 的表达式，就可以在相应的角度区间里进行积分。
- 由对称性不妨设 $p_0 \le q_0$，$p_1 \le q_1$，二元组 $(p_0, -q_0)$ 的字典序不大于 $(p_1, -q_1)$。
- 接下来要对区间 $[p_0, q_0]$ 和 $[p_1, q_1]$ 的相对位置关系进行讨论。

J. Computational Intelligence（1a. 区间相离）

- $q_0 \le p_1$
- 此时 $(1-x)p_0 + xq_0 \le q_0 \le p_1 \le (1-y)p_1 + yq_1$，可以直接拆开绝对值得到

$$
\begin{aligned}
f(\theta) &= -\int_0^1\int_0^1 (((1-x)p_0 + xq_0) - ((1-y)p_1 + yq_1))\mathrm{d}x\mathrm{d}y \\
&= -\int_0^1 ((1-x)p_0 + xq_0)\mathrm{d}x + \int_0^1 ((1-y)p_1 + yq_1)\mathrm{d}y \\
&= \frac{-p_0 - q_0 + p_1 + q_1}{2}
\end{aligned}
$$

J. Computational Intelligence（1b. 区间包含）

- $q_0 \geq q_1$
- 此时 $(1-x)p_0 + xq_0 \leq (1-y)p_1 + yq_1$ 成立当且仅当 (x, y) 在 $(0,0)$, $(x_0, 0)$, $(x_1, 1)$ 和 $(0,1)$ 围成的梯形区域 D 内，其中

$$x_0 = \frac{p_1 - p_0}{q_0 - p_0}, x_1 = \frac{q_1 - p_0}{q_0 - p_0}$$

- 记 $S = -\iint_{(x,y)\in D}(((1-x)p_0 + xq_0) - ((1-y)p_1 + yq_1))\mathrm{d}x\mathrm{d}y$
- 有

$$f(\theta) = 2S - \frac{-p_0 - q_0 + p_1 + q_1}{2}$$

J. Computational Intelligence（1b. 区间包含）

- 另一方面，不难得到 D 的面积为 $\frac{x_0+x_1}{2}$，重心坐标为 $\left(\frac{x_0^2+x_0x_1+x_1^2}{3(x_0+x_1)}, \frac{x_0+2x_1}{3(x_0+x_1)}\right)$。
- 结合面积和重心坐标的积分公式可知

$$\iint_{(x,y)\in D} \mathrm{d}x\mathrm{d}y = \frac{x_0+x_1}{2}$$

$$\iint_{(x,y)\in D} x\mathrm{d}x\mathrm{d}y = \frac{x_0^2+x_0x_1+x_1^2}{6}$$

$$\iint_{(x,y)\in D} y\mathrm{d}x\mathrm{d}y = \frac{x_0+2x_1}{6}$$

将以上三式代入 S 即可。

J. Computational Intelligence（1c. 区间相交）

- $p_1 < q_0 < q_1$
- 此时 $(1-x)p_0 + xq_0 \geq (1-y)p_1 + yq_1$ 成立当且仅当 (x, y) 在 $(1, 0)$，$(1, y_0)$ 和 $(1 - x_0, 0)$ 围成的三角形区域 D 内，其中

$$x_0 = \frac{q_0 - p_1}{q_0 - p_0}, x_1 = \frac{q_0 - p_1}{q_1 - p_1}$$

- 记 $S = \iint_{(x,y)\in D} (((1-x)p_0 + xq_0) - ((1-y)p_1 + yq_1))\mathrm{d}x\mathrm{d}y$，则有

$$f(\theta) = 2S + \frac{-p_0 - q_0 + p_1 + q_1}{2}$$

J. Computational Intelligence（1c. 区间相交）

- 另一方面，不难得到 D 的面积为 $\frac{x_0 y_0}{2}$，重心坐标为 $(1-\frac{x_0}{3}, \frac{y_0}{3})$。
- 结合面积和重心坐标的积分公式可知

$$\iint_{(x,y)\in D} \mathrm{d}x\mathrm{d}y = \frac{x_0 y_0}{2}$$

$$\iint_{(x,y)\in D} x\mathrm{d}x\mathrm{d}y = \frac{3x_0 y_0 - x_0^2 y_0}{6}$$

$$\iint_{(x,y)\in D} y\mathrm{d}x\mathrm{d}y = \frac{x_0 y_0^2}{6}$$

- 将以上三式代入 S 即可。

- 由于 p_0, q_0, p_1, q_1 是关于 $\cos(\theta)$ 和 $\sin(\theta)$ 的线性组合，在分别整理上述三种情况得到的表达式之后，除了较为简单的积分 $\int \cos(\theta)\mathrm{d}\theta$ 和 $\int \sin(\theta)\mathrm{d}\theta$ 之外，还需要计算两类复杂积分：

$$I_A : \int \frac{\cos(\theta-\alpha)\cos(\theta-\beta)}{\cos(\theta)}\mathrm{d}\theta$$

$$I_B : \int \frac{\cos^3(\theta-\beta)}{\cos(\theta)\cos(\theta-\alpha)}\mathrm{d}\theta$$

J. Computational Intelligence（2a. 计算积分 I_A）

- 这类积分由区间包含的情况导出。
- 利用 $\cos(\theta-\gamma)=\cos(\theta)\cos(\gamma)+\sin(\theta)\sin(\gamma)$ 展开分子可得

$$\int \frac{c_0\cos^2(\theta)+c_1\cos(\theta)\sin(\theta)+c_2\sin^2(\theta)}{\cos(\theta)}\mathrm{d}\theta$$

- 其中 c_0, c_1, c_2 均是关于 α 和 β 的常数。
- 整理约分后需要额外计算如下积分：

$$\begin{aligned}\int \frac{\sin^2(\theta)}{\cos(\theta)}\mathrm{d}\theta &= \int \frac{\mathrm{d}\theta}{\cos(\theta)} - \int \cos(\theta)\mathrm{d}\theta \\ &= \ln\left|\frac{\cos(\theta/2)+\sin(\theta/2)}{\cos(\theta/2)-\sin(\theta/2)}\right| - \sin(\theta) + C\end{aligned}$$

- 这里不使用常见形式 $\int \frac{\mathrm{d}\theta}{\cos(\theta)} = \ln|1/\cos(\theta)+\tan(\theta)|+C$ 是为了避免在可去间断点 $2\pi n-\frac{\pi}{2}(n\in\mathbb{Z})$ 处算出 `NaN` 的情况。

J. Computational Intelligence（2b. 计算积分 I_B，Case α）

- 这类积分由区间相交的情况导出，需要分情况讨论。
- 如果 $\sin(\alpha) = 0$，此时 $\cos(\theta - \alpha) = \cos(\theta)\cos(\alpha)$，于是有

$$\int \frac{\cos^3(\theta - \beta)}{\cos(\theta)\cos(\theta - \alpha)} \mathrm{d}\theta = \frac{1}{\cos(\alpha)} \int \frac{\cos^3(\theta - \beta)}{\cos^2(\theta)} \mathrm{d}\theta$$

- 仍然展开分子，整理约分后需要额外计算如下积分：

$$\begin{aligned}\int \frac{\sin^3(\theta)}{\cos^2(\theta)} \mathrm{d}\theta &= \int \frac{\sin(\theta)}{\cos^2(\theta)} \mathrm{d}\theta - \int \sin(\theta) \mathrm{d}\theta \\ &= 1/\cos(\theta) + \cos(\theta) + C\end{aligned}$$

- 否则 $\sin(\alpha) \neq 0$，展开分子可得

$$\int \frac{c_0 \cos^3(\theta) + c_1 \cos^2(\theta)\sin(\theta) + c_2 \cos(\theta)\sin^2(\theta) + c_3 \sin^3(\theta)}{\cos(\theta)\cos(\theta - \alpha)} \mathrm{d}\theta$$

- 其中 c_0, c_1, c_2, c_3 均是关于 β 的常数，分子中包含 $\cos(\theta)$ 的项均能通过作代换 $\theta \to \theta + \alpha$ 转化成上述已讨论的情况，但是仍需要计算如下积分：

$$\int \frac{\sin^3(\theta)}{\cos(\theta)\cos(\theta - \alpha)} \mathrm{d}\theta$$

J. Computational Intelligence（2b. 计算积分 I_B，Case β）

- 注意到 $\sin(\theta) = \frac{1}{\sin(\alpha)}(\cos(\theta-\alpha) - \cos(\theta)\cos(\alpha))$，于是有

$$
\begin{aligned}
&\int \frac{\sin^3(\theta)}{\cos(\theta)\cos(\theta-\alpha)} \mathrm{d}\theta \\
=& \frac{1}{\sin(\alpha)} \int \frac{\sin^2(\theta)}{\cos(\theta)} \mathrm{d}\theta - \frac{\cos(\alpha)}{\sin(\alpha)} \int \frac{\sin^2(\theta)}{\cos(\theta-\alpha)} \mathrm{d}\theta
\end{aligned}
$$

- 这里前一项是上述已讨论的情况，后一项通过作代换 $\theta \to \theta + \alpha$ 也能转化成上述已讨论的情况。

- 最后还要注意避免系数是 0 的积分结果发散导致算出 `NaN` 的情况。
- 实现时要用绝对值 $< \epsilon$ 替代所有 $= 0$ 的判断，这可能会导致额外的精度损失。
- 在现有测试数据下，基于上述做法的 `C++` 标程的输出结果与基于从 *WolframAlpha* 得到二重积分公式并使用较高精度进行较长时间计算的 `Python` 代码的输出结果相比，相对误差不超过 10^{-12}。
- 现场评测时使用了 `Python` 代码的输出结果作为标准输出。

最后

- 没听明白？没关系。
- 访问 https://sua.ac/wiki/ ，有文字版题解与带注释的参考代码。

Thank you!