第十三届山东省 ICPC 大学生程序设计竞赛

SUA 程序设计竞赛命题组

2023 年 6 月 4 日

概况

- 阶段一（基本编程技巧）：I（循环）、A（排序）、G（排序）、D（二分）。
-
- 阶段二（常见算法应用）：L（简单构造）、E（进制）、B（拓扑排序）、J（位运算）、M（几何）。
-
- 阶段三（高级算法应用）：K（思维 & 递推）、F（线段树优化 dp）、C（类后缀数组）。
-
- 阶段四（打星队娱乐用）：H。

I. Three Dice

题意

- 问是否能投掷出三只骰子，使得朝上的面的红色点数之和为 a，黑色点数之和为 b。

- 直接枚举三个骰子的点数判定即可。
- 时间复杂度为 $O(1)$。

A. Orders

题意

- 某工厂在第 1 天开工之前收到了 n 笔订单，第 i 笔订单可以用两个整数 a_i 和 b_i 描述，表示工厂需要在第 a_i 天结束时交付 b_i 件货物。
- 已知工厂每天能生产 k 件货物，且第 1 天开工之前没有任何存货，问该工厂能否完成所有订单。

- 将所有订单按 a_i 排序，检查排序后的订单 i 之前，存货量增加 $k \times (a_i - a_{i-1})$ 即可。
- 复杂度 $\mathcal{O}(n \log n)$，主要是排序的复杂度。

G. Matching

题意

- 给定长度为 n 的整数序列 $a_1, a_2, \cdots, a_n$，我们将从该序列中构造出一张无向图 G。
- 具体来说，对于所有 $1 \le i < j \le n$，若 $i - j = a_i - a_j$，则 G 中将存在一条连接节点 i 与 j 的无向边，其边权为 $(a_i + a_j)$。
- 求 G 的一个匹配，使得该匹配中所有边的边权之和最大，并输出最大边权之和。

G. Matching

- 移项得 $i - a_i = j - a_j$。因此所有 $(u - a_u)$ 为相同值的节点 u 组成一个团（任意两点之间两两都有连边的连通块）。
- 团与团之间因为没有连边，答案不互相影响，因此可以每个团单独计算答案并加起来。
- 由于每条边的边权实际上是两个端点的点权之和，因此对于一个团，每次选择点权最大的两个节点，看它们加起来是否为正数即可。
- 复杂度 $\mathcal{O}(n \log n)$，主要是给点权排序的复杂度。

D. Fast and Fat

题意

- 一场团体越野比赛有 n 名队员，其中第 i 名队员的速度为 v_i，体重为 w_i。
- 当队员 i 背着队员 j 时，如果队员 i 的体重大于等于队员 j，则队员 i 的移动速度不会变化；如果队员 i 的体重小于队员 j，则队员 i 的移动速度变为 $v_i - (w_j - w_i)$。
- 求所有未被背负的队员中最慢速度的最大值。

D. Fast and Fat

- 二分整个队伍的速度至少为 x，接下来考虑如何检验答案。
- 如果队员 i 的速度大于等于 x，则至多可以背起体重为 $(v_i - x + w_i)$ 的队员。如果队员 i 的速度小于 x，则必须被别人背负。因此问题可以转化为：

> 有 p 个人和 q 件工作，第 i 个人的能力值为 a_i，第 i 项工作的难度为 b_i，只有 $a_i \ge b_j$ 才能让第 i 个人做第 j 项工作。每个人最多做一项工作，问所有工作是否都能被完成。

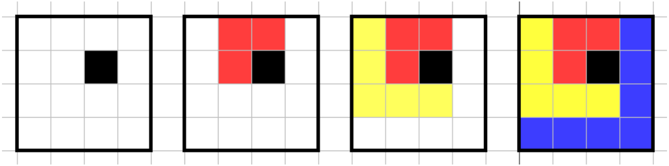
- 这是一个经典的贪心问题。首先选出能力值最高的 q 个人，然后将第 i 难的工作派给能力值第 i 高的人即可。
- 如果预先将队员分别按 $(v_i + w_i)$ 以及 w_i 排序，本题的复杂度可以做到 $\mathcal{O}(n \log n + n \log(\max v_i))$。

题意

- 给定一个 n 行 n 列的网格，网格中包含恰好一个黑色方格，其余方格均为白色。令 (i, j) 表示位于第 i 行第 j 列的格子，这个黑色方格位于 (b_i, b_j)。
- 您需要用若干 L 形覆盖所有白色格子，使得每个白色格子都恰好被一个 L 形所覆盖，同时唯一的黑色方格不能被任何 L 形覆盖。L 形不能超过网格的边界。

L. Puzzle: Sashigane

- 我们从黑色格子开始，每次操作向外“包”一层 L，使得第 i 次操作结束后能形成一个 $(i+1)\times(i+1)$ 的正方形。如下图所示。



- 复杂度 $\mathcal{O}(n)$。

E. Math Problem

题意

- 给定 n 和 k，可以进行以下两种操作任意次：
  - 选择一个整数 x 满足 $0 \le x < k$，将 n 变为 $k \cdot n + x$。该操作每次花费 a 枚金币。每次选择的整数 x 可以不同。
  - 将 n 变为 $\lfloor \frac{n}{k} \rfloor$。该操作每次花费 b 枚金币。其中 $\lfloor \frac{n}{k} \rfloor$ 表示小于等于 $\frac{n}{k}$ 的最大整数。
- 给定正整数 m，求将 n 变为 m 的倍数最少需要花费几枚金币。

E. Math Problem

- 显然一定先进行除法，然后再进行乘法，不会把新乘上去的东西再除掉。
- 进行 p 次操作后，n 的范围是 $[k^p \times n, k^p \times (n+1) - 1]$，只要这个范围里面包括 m 的倍数即可停止乘法操作。因此乘法操作至多进行 $\log_k m$ 次。
- 所以我们枚举除法操作进行几次，然后枚举乘法操作进行几次即可。复杂度 $\mathcal{O}(\log_k n \times \log_k m)$。

B. Building Company

题意

- 公司有 g 类员工，每一类员工都属于一个工种。第 i 类员工的工种编号为 t_i，共有 u_i 人。
- 有 n 项工程等待承接。承接第 i 项工程需要满足 m_i 项要求，其中第 j 项要求至少有工种编号为 $a_{i,j}$ 的员工 $b_{i,j}$ 人。
- 承接该工程后，会吸引 k_i 类员工加入公司，其中第 j 类员工的工种编号为 $c_{i,j}$，共有 $d_{i,j}$ 人。
- 求最多能承接多少工程。

B. Building Company

- 本题类似于拓扑排序。
- 我们维护每项工程还有几条要求没有满足，并将不同工种的要求分别维护。所有要求都被满足的工程将加入队列。
- 从队列中取出一项工程后，我们会获得该工程的奖励。当工种为 t 的员工加入公司后，我们检查和工种 t 有关的人数最少的未满足需求。如果这项需求被满足了，则对应工程的要求数减一。要求数减少为零的工程继续加入队列。
- 答案就是从队列中取出的工程数。复杂度 $\mathcal{O}(n\log n)$，主要是每次拿出“人数最少的需求”需要预先排序或使用堆等数据结构。

J. Not Another Path Query Problem

题意

- 给定一张无向图，每条边有边权，以及一个整数阈值 V。q 次询问，每次给定 x, y，你需要判定是否存在一条从 x 到 y 的边权 AND 和大于等于 V 的路径。

J. Not Another Path Query Problem

- 考虑如果一个整数 w 大于 V，我们枚举其二进制表示与 V 的 LCP（最长公共前缀）。
- 那么，如果它的下一位为 1，那么 w 的下一位也必须为 1，该 LCP 长度不合法。如果下一位为 0，则当 w 下一位为 1 时，更低的位可以随意取。
- 由于 LCP 只有 $O(\log V)$ 种，因此我们可以枚举 LCP 的长度，此时我们便可以判定哪些边可以留在图中（即 LCP 中为 1 的位置必须为 1，其余位置无任何限制）。
- 在加入所有边后可以使用并查集判定连通性，时间复杂度为 $O(m \log V\alpha(n))$。
- 当然，由于我们所有的操作只有加边后查询，因此并查集也是不必要的。可以 BFS 出每个联通块的编号即可。时间复杂度为 $O(m \log V)$。

M. Computational Geometry

题意

- 给定一个有 n 个顶点的凸多边形 P，您需要选择 P 的三个顶点，按逆时针顺序记为 a，b 和 c。要求在 b 沿逆时针方向到 c 之间恰有 k 条边。
- 考虑用线段 ab 和 ac 将 P 割开。将由线段 ab，ac，以及 b 和 c 之间的 k 条边围成的 $(k+2)$ 边形记作 Q。
- 求 Q 可能的最大面积。

M. Computational Geometry

- 考虑固定边 bc 后，多边形可以分为两部分，第一部分为 bc 所在的 $(k+1)$ 边形，第二部分则为 abc 构成的三角形。
- 注意到 b 固定后 c 是唯一确定的，也就是一共只有 $O(n)$ 种 bc。
- 枚举 bc，第一部分的面积可以预处理出来，而 a 的选择不会影响第一部分的面积。
- 问题变为找到 a 使得 $\triangle abc$ 的面积最大。
- 有多种方法可以解决该问题。可以每次询问时三分找到点 a，也可以使用双指针。
- 时间复杂度为 $O(n \log n)$ 或 $O(n)$，均可轻易通过此题。

K. Difficult Constructive Problem

题意

- 给定一个长度为 n 的字符串 $s_1s_2\cdots s_n$，其中 $s_i \in \{\text{‘0’}, \text{‘1’}, \text{‘?’}\}$
- 另外给定一个整数 k，请将字符串中所有的 ‘?’ 换成 ‘0’ 或 ‘1’，使得满足 $1 \le i < n$ 且 $s_i \ne s_{i+1}$ 的下标 i 恰有 k 个。
- 不同的 ‘?’ 可以用不同字符替换。
- 求字典序最小的一组解。

K. Difficult Constructive Problem

- 假设字符串头尾不存在问号，只有中间有问号。
- 可以发现，当把一个字符从 0 变成 1，或者从 1 变成 0 后，满足条件的下标数量将会增加或减少 2。
- 因此满足条件的下标数量在满足奇偶性的前提下，是可以取到“连续”值的。
- 因此计算 $f(i, 0/1)$ 以及 $g(i, 0/1)$，表示只考虑从第 i 个字符开始的后缀，且第 i 个字符填 0 或 1 时，后缀中最少（最多）有几个满足条件的下标。转移方程为

$$f(i,j) = \min(f(i+1,j), f(i+1,1-j)+1)$$

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

K. Difficult Constructive Problem

- 这样我们就可以从头开始逐位确定答案。
- 先看 $f(1, s_1)$ 和 k 的奇偶性是否一样，对于每一位再看是否 $f(i, 0/1) \le k - k' \le g(i, 0/1)$，其中 k' 表示前 i 个字符中满足条件的下标数量。
- 最后考虑问号在字符串头尾的情况。
- 其实只要枚举头尾的两个字符是 0 还是 1 即可。
- 复杂度 $\mathcal{O}(n)$。

F. Colorful Segments

题意

- 有 n 条线段，第 i 条线段的颜色为 c_i $(0 \le c_i \le 1)$。
- 您需要选择若干条线段。
- 如果选择的两条线段有重合，则这两条线段的颜色必须相同。
- 求选择线段的不同方案数。

F. Colorful Segments

- 首先将所有线段按右端点排序，我们把所选线段分成若干连续段，每一段内的线段颜色都相同。
- 令 $f(i,c)$ 表示只考虑前 i 条线段，且第 i 条线段必选，且第 i 条线段的颜色为 c 的方案数。转移时，我们枚举上一段的终点 j（j 的颜色必须和 i 不同），则转移方程为

$$f(i,c) = \sum_{j=1}^{p_i} f(j, 1-c) \times 2^{g(r_j+1, r_i, c)-1}$$

- 其中 p_i 表示满足 $r_{p_i} < l_i$ 的最大下标，$g(r_j+1, r_i, c)$ 表示被完全包含在 $[r_j+1, r_i]$ 这段区间内，且颜色为 c 的线段数量。
- 直接计算这个 dp 的复杂度是 $\mathcal{O}(n^2)$ 的，考虑用数据结构优化。
- 当我们加入第 i 条线段时，所有满足 $1 \le j \le p_i$ 的贡献都会乘以 2。因此我们使用线段树维护前缀乘 2，单点赋值，以及前缀求和即可。复杂度 $\mathcal{O}(n \log n)$。

C. Trie

题意

- 给定一棵有 $(n+1)$ 个节点的有根树。
- 树上共有 m 个关键节点，其中第 i 个关键节点的编号为 k_i。保证所有叶子节点都是关键节点。
- 请为每一条边标上一个小写字母，使得这棵有根树变为一棵大小为 $(n+1)$ 的字典树。
- 考虑所有关键节点代表的字符串构成的序列 $A = \{s(k_1), s(k_2), \cdots, s(k_m)\}$，设 $B = \{w_1, w_2, \cdots, w_m\}$ 是由序列 A 中所有字符串按字典序从小到大排序后得到的字符串序列，您需要找到一个标记字母的方案，使得序列 B 最小。

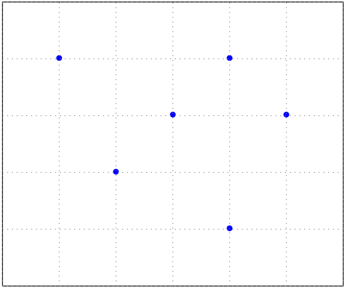
C. Trie

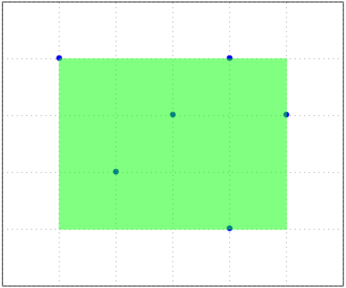
- 本题思路类似于后缀数组的计算。
- 对于同一个节点，考虑如何将字母分给它的所有子树。显然，a 应该分给最小的子树，b 应该分给第二小的子树，…
- 这里，子树 i 小于子树 j，指的是子树 i 里最小的关键字符串比子树 j 里的小，如果一样就比第二小的关键字符串…
- 如果某个子树里的字符串已经比完了，那么哪个子树有更多的字符串，哪个子树就更小。
- 假设对于每棵子树，我们已经知道一个值 $rank[i]$，$rank[i]$ 越小的子树越小。因为同一层的子树会在它们的最近公共祖先进行比较，因此我们需要按层给子树算 rank。
- 子树的排序依据是：把根节点的所有子节点的 rank 放进一个 'vector' 里，把该 'vector' 排序，比较 'vector' 的大小即可。注意，当一个 'vector' 是另一个 'vector' 的前缀时，更长的 'vector' 应该排在更前面。
- 复杂度 $\mathcal{O}(n \log n)$。

题意

- 在一个 $n \times m$ 的矩形里有 k 个特殊点，称一个正方形是好的，当且仅当其完全位于该矩形内，且其不严格覆盖任意一个特殊点。
- 求所有好的矩形的面积之和。

- 我们考虑使用容斥原理，如果我们对每个点集的子集 S，计算出所有覆盖了 S 内所有点的方案数 $f(S)$，那么答案即为 $\sum_{S}(-1)^{|S|}f(S)$。
- 注意到，对于一个集合 S，完整覆盖 S 内所有点等价于覆盖 S 的包围盒。





H. Be Careful 2

- 注意到子集的数量会有 $\mathcal{O}(2^k)$ 个，但不同的包围盒数量至多只有 $\mathcal{O}(k^4)$ 个。
- 考虑转而枚举所有的包围盒 B，计算出所有包围盒为 B 的 S 的 $(-1)^{|S|}$ 之 $g(B)$，则答案即为 $\sum_B f(B)g(B)$。
- 注意到，如果包围盒内部（不含边界）内有任意一个点，则该包围盒的容斥系数之和一定为 0。
- 因此我们只需要找到所有内部不包含任何点的包围盒即可。

H. Be Careful 2

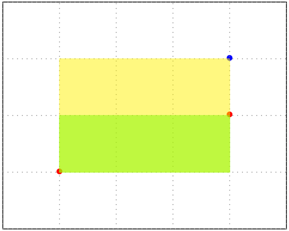
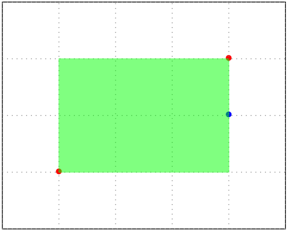
- 考虑如果所有点的横坐标互不相同，我们应当如何处理。
- 考虑枚举包围盒左边界的点 (x_l, y_l) 与包围盒右边界的点 (x_r, y_r)，则注意到：
  1. 对所有 $x \in (x_l, x_r), y \in [y_l, y_r]$，点 (x, y) 均不能出现。
  2. 对所有 $x \in (x_l, x_r)$ 的点，找到 y_l 的前驱与 y_r 的后继，这些点是唯一可能作为包围盒上下边界的点
- 因此对于一对 (l, r)，只有至多四个对应的包围盒。
- 如何定位到这些包围盒？我们枚举 x_l 后，按照 x_r 从大到小的顺序枚举右边界上的点，使用链表维护每组纵坐标点前驱后继，即可快速定位到对应的点。
- 因此寻找这些包围盒的时间复杂度为 $O(k^2)$。

H. Be Careful 2

- 考虑如果所有点的横坐标可以相同，我们应当如何处理。
- 我们同样把所有点按照横坐标排序，对于横坐标相同的点，我们按照纵坐标对其排序。
- 容易发现上述性质仍然是成立的，我们在枚举到一组点时，只需要取出 (x_l, x_r) 对应的点即可。
- 为什么？我们考虑所有枚举出的情况。

H. Be Careful 2

- 我们以如下情况为例

- 我们会在枚举左图中的红点为左右边界点时将其计入 -1 的贡献，而在枚举右图的红点时将其计入 +1 的贡献，因此贡献之和为 0。

- 因此我们可以在 $O(k^2)$ 的时间复杂度内找到所有的包围盒。
- 对于每个形如 $(x_1, y_1) - (x_2, y_2)$ 的包围盒，其贡献可表示为:

$$\sum_d d^2 \cdot \omega_x(d) \cdot \omega_y(d)$$

- 其中

$$\omega_x(d) = \max(0, \min(x_0 - 1, n - d) - \max(0, x_1 - d - 1) + 1)$$

$$\omega_y(d) = \max(0, \min(y_0 - 1, m - d) - \max(0, y_1 - d - 1) + 1)$$

- 注意到这可以表示成若干个多项式组成的分段函数，每个多项式的次数不超过 4。
- 因此我们找到所有的分段点，每一段只需要求一个不超过四次点多项式的前缀和，这是容易的。
- 因此我们可以 $O(1)$ 计算贡献，总的时间复杂度为 $O(k^2)$。

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!