2023 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

2023 年 11 月 5 日

概况

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

I. Counter

题意

- 有一个计数器，一开始是 0，每次操作可以把值 $+1$ 或清零。给若干已知条件表示第 x_i 次操作后计数器的值为 v_i，问是否有一种操作序列满足所有的已知条件。

- 如果第 x_i 次操作后计数器的值为 v_i，说明第 $(v_i - x_i)$ 次操作是清零，且 $[v_i - x_i + 1, x_i]$ 这个操作区间内没有清零操作。
- 把区间排序后进行判断即可。复杂度 $\mathcal{O}(n \log n)$。

F. Equivalent Rewriting

题意

- 有 n 个操作和 m 个变量，初始值均为 0。
- 现在按 $1 \sim n$ 的顺序执行这些操作，第 i 个操作会将其中 p_i 个变量赋值为 i。
- 求一种不同于 $1 \sim n$ 的执行顺序 q，使得按照 $q_1, q_2, \ldots, q_n$ 的顺序执行操作会和原顺序得到一样的最终结果。

F. Equivalent Rewriting

- 考虑将原操作序列建模成一个有向无环图 G，第 i 个操作向第 j 个操作有一条连边 $i \to j$ 当且仅当存在某个变量 t 使得 j 是在原顺序中修改 t 的最后的操作，且 i 也修改了 t。
- 可以证明，G 的任意一个拓扑序都会和原序列产生一样的操作结果。因为这样的建图保证了对于每个变量，在任意一种拓扑序中最后一次修改它的操作一定是原序列中最后一次修改它的操作。而最终序列的结果之和每个变量最后被哪个操作修改有关，得证 G 的任意一个拓扑序都是合法的。

F. Equivalent Rewriting

- 现在问题转化为，给定一张有向无环图 G 和一个拓扑序 $1 \sim n$，求任意一个不同的拓扑序或告知无解。
- 一种简单的做法是考虑尝试交换 $1 \sim n$ 的相邻两个元素，如果交换一次任意两个相邻元素都无法得到新的合法拓扑序，说明 G 依赖关系至少是一条 $1 \sim n$ 的链，即拓扑序唯一。
- 我们并不需要将 G 显式的建立出来，只要枚举并判断操作 i 和操作 $i+1$ 之间是否有连边，如果没有，则找到了一组解：

$$\forall 1 \le j \le n, q_j = \begin{cases} i, & j = i+1 \\ i+1, & j = i \\ j, & \text{otherwise} \end{cases}$$

- 单组数据的时间复杂度是 $\mathcal{O}(n + \sum_{i=1}^{n} p_i)$。

C. Primitive Root

题意

- 给定质数 P 和非负整数 m，有多少非负整数 g 满足 $g \leq m$ 且 $g \oplus (P-1) \equiv 1 \pmod{P}$？这里 $\oplus$ 是按位异或（XOR）运算符。

C. Primitive Root

- 解法 1：
  $[0, m]$ 区间内的数异或上一个固定的数 x 会划分成 $\log(m)$ 段区间，计算出每段区间有多少个形如 $kP+1$ 的数即可。
- 解法 2：
  设 $g \oplus (P-1) = kP+1$，则需要统计有多少个 $k \geq 0$ 满足 $(kP+1) \oplus (P-1) = g \leq m$。
  注意到异或的性质：$a - b \leq a \oplus b \leq a + b$。
  $\forall\ 0 \leq k \leq \lfloor \frac{m}{P} \rfloor - 1$，均有 $(kP+1) \oplus (P-1) \leq (k+1)P \leq m$。
  $\forall\ k \geq \lceil \frac{m}{P} \rceil + 1$，均有 $(kP+1) \oplus (P-1) \geq (k-1)P+2 > m$。
  所以我们只需要检查中间 $O(1)$ 个 k 是否满足 $(kP+1) \oplus (P-1) \leq m$ 即可。

G. Knapsack

题意

- 给定 n 件物品的体积 w_i 与价值 v_i，以及背包的总容量 W。
- 你可以优先选择 k 件物品获得他们的价值，随后再选择一些物品，要求随后选择的物品的体积之和不超过 W。
- 最大化选择物品的价值之和。

G. Knapsack

- 注意到以下两件事实：
  1. 如果存在物品 a, b 满足 a 被免费选走，b 被付费选走，那么必有 $w_a \geq w_b$。
  2. 如果存在物品 a, b 满足 a 被免费选走，b 没有被选走，那么必有 $v_a \geq v_b$。
- 若不然，我们可以选择将 a 所占用的免费名额替换为 b，方案不会变劣。
- 因此，如果我们将所有物品按照 w_i 从大到小排序，那么对于最优策略而言，一定存在一个分界点 M，满足 $i \leq M$ 的物品中，价值前 k 大的物品被免费选走。
- 对于每个 i，可以通过维护一个堆来预处理出前 i 个物品被免费选走的物品的价值之和。
- 因此我们只需要对每个后缀维护出 $0/1$ 背包的结果即可。
- 时间复杂度为 $\mathcal{O}(nW + nk + n \log n)$.

A. Cool, It's Yesterday Four Times More

题意

- 给一张网格图，每个格子要么是洞，要么是空地。一开始每个空地都有一只袋鼠。
- 每次操作可以让所有袋鼠同时往上/下/左/右移动一步，走出网格或走到洞里的袋鼠就被移除。如果最后恰好剩余一只袋鼠，它就是赢家。求可能成为赢家的袋鼠数量。

- 维护 (i, j, i', j') 表示考虑位于 (i, j) 的袋鼠最后能否自己存活，且把位于 (i', j') 的袋鼠移除。对于固定的 (i, j)，只要所有的 (i, j, i', j') 都是 true，那么 (i, j) 就可以是赢家。
- (i, j, i', j') 可以从 $(i \pm 1, j \pm 1, i' \pm 1, j' \pm 1)$ 转移而来，可以通过一次 dfs 或 bfs 求出所有的值。

A. Cool, It's Yesterday Four Times More

- 您可能会担心：会不会一开始 (i, j, i_1, j_1) 是 true，结果 (i, j) 经过一些操作把 (i_2, j_2) 移除之后，(i_1, j_1) 变得无法移除了？
- 这是不会的，因为两只袋鼠的相对位置都是不变的，因此每次操作都可以撤销，可以把 (i, j) 和 (i_1, j_1) 都回到原来的位置。
- 复杂度 $\mathcal{O}((nm)^2)$。

L. Elevator

题意

- 给一个序列 $(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)$，将序列中的所有数对分成若干组，要求每组 $\sum b_i \leq k$。一个分组方案的得分是所有组 $\max a_i$ 的和，求最小得分。

- 我们可以使用如下贪心策略：将所有包裹（无论重量是多少）按目标楼层从高到低排序，之后每趟电梯按该顺序枚举所有还未被运送的包裹并加入电梯，直到电梯装满或所有包裹都被枚举。枚举过程中可能会出现电梯容量只剩 1，但是下一个包裹的重量却是 2 的情况。此时要继续枚举，直到出现重量为 1 的包裹。

L. Elevator

- 我们通过说明该贪心策略产生的答案达到了下界来说明最优性。为了求出答案的下界，我们将所有重量为 2 的包裹拆成两个重量为 1 的包裹，变成这样一个新问题：

新问题

- 给一个序列 $a_1, a_2, \cdots, a_n$，将序列中的所有数分成若干组，每组最多包含 k 个数。一个分组方案的得分是所有组最大值的和，求最小得分。

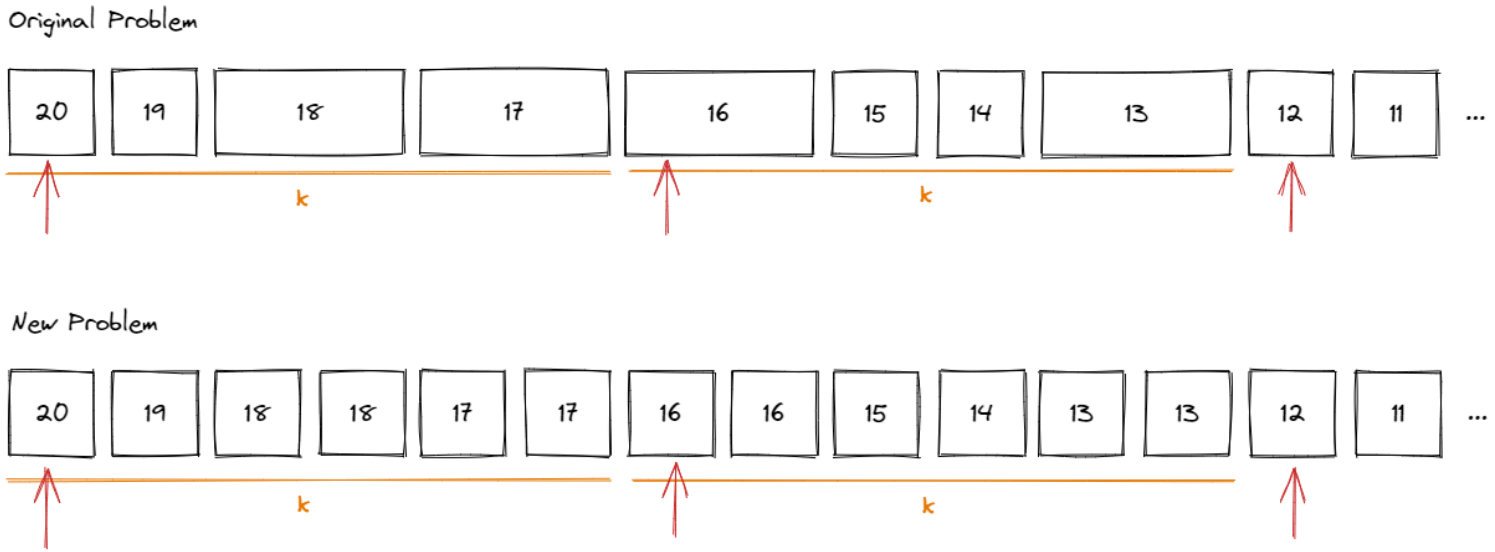
- 由于原问题中的任何一个配送方案都能对应新问题中的一个配送方案，因此新问题的答案不比原问题大，也就是说新问题的答案就是原问题答案的下界。

L. Elevator

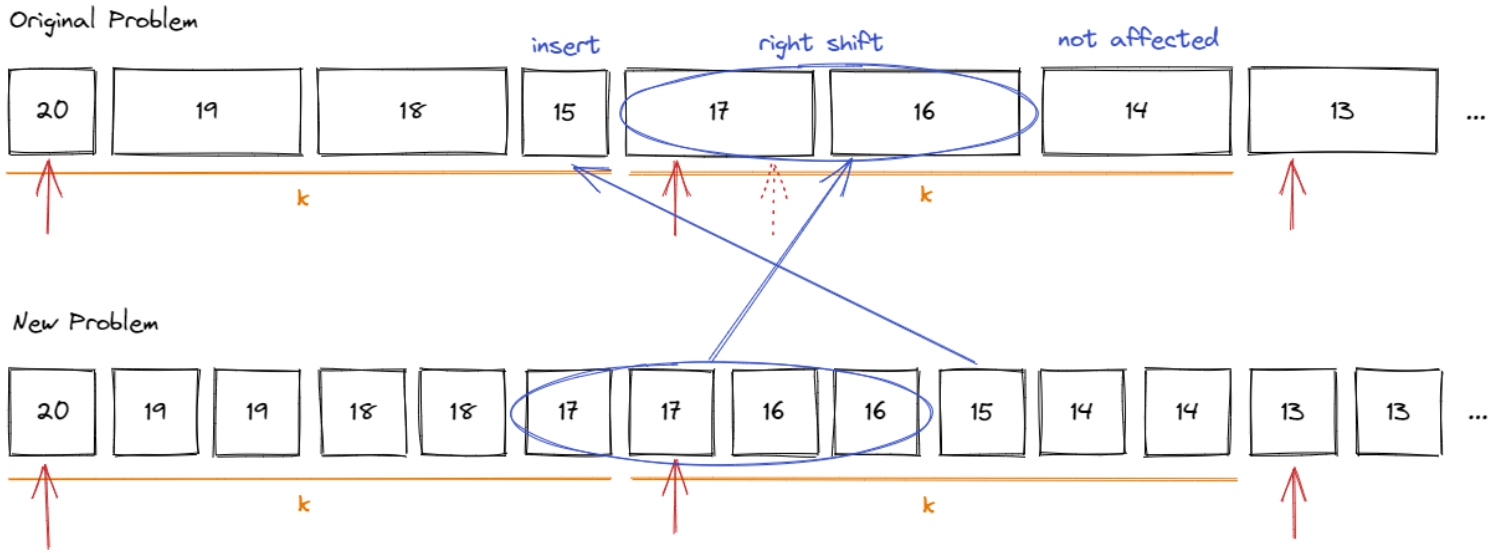
- 这个新问题是一个经典问题，我们只要把序列从大到小排序，然后把第 1 到 k 个数分成一组，第 $(k+1)$ 到 $2k$ 个数分成一组... 即可。答案就是所有下标对 k 取模等于 1 的数的和。
- 接下来证明原问题的贪心策略也能得到这个答案。分为两种情况。在示意图中，我们用红色箭头指向每趟电梯的最大楼层，也就是真正消耗电量的包裹。

L. Elevator

- 前面总重量为 k 的货物恰好能装满一趟电梯。此时原问题和新问题消耗了相同的电能。



- 出现电梯容量只剩 1，但是下一个包裹的重量却是 2 的情况。



- 由于 k 是偶数，因此在下一个重量为 1 的包裹出现之前，新问题的红箭头指向的都是大包裹的后半部分。在原问题的贪心策略中，我们把下一个重量为 1 的包裹移到了前面，那么中间的所有大包裹都会右移一个位置，红箭头也从大包裹的后半部分移到了前半部分。但所有红箭头指向的仍然是同一个包裹，因此答案也没有改变。

L. Elevator

- 根据证明过程，原问题和新问题的答案是一样的。因此我们只要求新问题的答案即可。由于包裹总数可能很多，我们要用除法快速计算目的地相同的包裹对答案的影响。
- 总复杂度 $\mathcal{O}(n \log n)$，主要是排序的复杂度。

M. Trapping Rain Water

题意

- 给定序列 $a_1, a_2, \cdots, a_n$，第 i 次操作会将 a_{x_i} 的值增加 v_i。设 f_i 表示前缀最大值，g_i 表示后缀最大值，每次操作后求 $\sum_{i=1}^{n}(\min(f_i, g_i) - a_i)$。

- 注意到 $\min(f_i, g_i)$ 是一个山峰形状的序列，观察可知 $\min(f_i, g_i) = \sum f_i + \sum g_i - \max a_i$。
- 用 set 维护 f_i 形成的单调栈。每次操作后，会往 set 里插入至多一个数，然后根据单调栈的性质把右边比它小的数都踢掉。$\sum f_i$ 就用 f_i 乘以区间长度的方式算出。g_i 的单调栈可以用相似的方式维护。
- 每个数最多被踢掉一次，因此复杂度 $\mathcal{O}(n \log n)$。

D. Red Black Tree

题意

- 红黑树要求每个点到任意后代叶子节点的路径上，黑色点的数量都相同。称该性质为“红黑树性质”。
- 现在给定一棵树，每个点是红色或黑色，对于所有 $k \in [1, n]$，求为了让以节点 k 为根的子树满足红黑树性质，至少要修改几个点的颜色。

- 设 $f(u, x)$ 表示以 u 为根的子树满足红黑树性质，且从 u 到任意后代叶子节点的路径上都有 x 个黑色点需要的最少修改次数。有朴素的 dp 方程：

$$f(u, x) = \min_{i \in [0,1]} \left(g(u, i) + \sum_{v \in \mathrm{son}(u)} f(v, x - i) \right)$$

- $g(u, 0/1)$ 是让节点 u 变红/黑的代价。

D. Red Black Tree

- 可以归纳证明 $f(u,x)$ 是关于 x 的凸序列，因为 $g(u,0/1)$ 是凸序列（只有两个点），凸序列的和还是凸序列，凸序列的 $(\min,+)$ 卷积也还是凸序列。
- 凸序列常用单调的差分数组进行维护。由于 x 的取值范围是子树深度的 min，因此 $\sum$ 部分直接暴力把子树的差分数组加起来即可，复杂度是 $\mathcal{O}(n)$ 的。
- 而 $g(u,1)-g(u,0)=\pm 1$，因此我们还要支持往差分数组里插入一个 1 或 -1，并维持差分数组的单调性。
- 可以用两个 vector，一个 vector 保存所有负数，一个 vector 保存所有正数，再开一个变量记录有几个 0。这样插入 1 就往正数 vector 的开头插，插入 -1 就往负数 vector 的末尾插。答案就是 $f(u,0)$ 加上差分数组的最小前缀和，即负数 vector 的元素之和。这样复杂度仍然是 $\mathcal{O}(n)$ 的。

K. Grand Finale

题意

- 给定 n 张起始手牌，和一个按顺序排列好的大小为 m 的牌堆，卡牌由 $[0]$, $[1]$, $[2]$ 三种组成。
- 其中 $[0]$ 表示不能打出，$[1]$ 表示打出后能从牌堆中抽 1 张牌，$[2]$ 表示打出后能从牌堆中抽 2 张牌。
- 存在手牌上限 k，当在手牌数量等于手牌上限时打出一张 $[2]$，会直接摧毁抽上来的第二张牌而不是置入手牌。
- 求最小的 k 使得在最优策略下能够将牌库中的所有牌抽完，或告知不可能。

K. Grand Finale

我们可以观察得到如下结论：

1. 手牌数量随着游戏进行是单调不降的。
2. 如果有最小可行的 k，那么决策的过程中一定会有一个时刻手牌数量到达了上限，否则 k 还可以更小。
3. 手牌上限 k 对于游戏的影响仅限于改变到达满手牌状态的时刻，并且一旦到达了满手牌状态，无论怎么决策之后的状态都会是满手牌。

K. Grand Finale

- 根据以上结论可知，最优解一定会进入满手牌状态，且进入满手牌状态以后，所有的决策都不再受 k 的大小的影响。
- 因而对于进入满手牌状态后的最优决策，我们可以利用动态规划求解。
- 不妨令 $f(i,j)$ 表示当前手牌已满，手上有 j 张 $[2]$，牌堆抽到了第 i 张牌，最少需要此时手上还有多少张 $[1]$ 才能抽完。此时转移可以通过枚举下一张使用的牌是 $[1]$ 还是 $[2]$ 来 $\mathcal{O}(1)$ 完成。

K. Grand Finale

- 在求出所有 $f(i,j)$ 的基础上，我们尝试判断一个 k 可不可行。
- 我们可以枚举第一次进入满手牌状态的时间 t，此时手上的 $[1]$ 的数量和 $[2]$ 的数量都是可以计算得出的。
- 这是因为这是第一次满手牌的时刻，所以在这之前一定恰好使用了 $k-n$ 张 $[2]$ 和 $t-(k-n)$ 张 $[1]$，并且 t 之前的牌堆里的所有牌都被抽进了手牌。
- 我们可以记牌堆中前 $t-1$ 张牌及起始手牌中共含有 sum_1 张 $[1]$ 和 sum_2 张 $[2]$，那么此时手牌中的 $[1]$ 数量 cnt_1 和 $[2]$ 数量 cnt_2 分别为：

$$cnt_1 = sum_1 - t + k - n, \quad cnt_2 = sum_2 - k + n$$

K. Grand Finale

- 此时我们可以通过判断 $f(t, cnt_2) \leq cnt_1$ 是否成立来验证 k 是否是一个可行的手牌上限。
- 但需要留意的是，即使在不触及手牌上限的情况，我们不一定能通过抽卡到达 $f(t, cnt_2)$ 这一个状态，或者这个状态本身就不合法 (即 $cnt_1 < 0$ 或者 $cnt_2 < 0$)。
- 对于后者我们在计算的时候判断一下边界即可。对于前者，我们需要一个额外的部分来判断能否在不触及手牌上限的情况下到达这个状态。

K. Grand Finale

- 这里有多种解决方法，其中一种思路是再次采用动态规划，令 $g(i,j)=0/1$ 表示手牌永远不会到达上限，当前手上有 j 张 $[2]$，牌堆抽到了第 i 张牌，是否可行。
- 因为不存在手牌上限，所以此时手上有多少张 $[1]$ 在 j 确定的情况下是可以计算得出的，所以这里同样可以通过枚举下一张使用的牌是 $[1]$ 还是 $[2]$ 来 $\mathcal{O}(1)$ 完成转移。
- 那么我们就完善了判断一组 (k,t) 是否合法的条件:

$$(cnt_1 \geq 0) \wedge (cnt_2 \geq 0) \wedge (f(t,cnt_2) \leq cnt_1) \wedge (g(t,cnt_2)=1)$$

- 单组数据的时间复杂度是 $\mathcal{O}(nm)$。

E. Extending Distance

题意

- 有一个 n 行 m 列的网格图，相邻格点之间有边，边有边权。
- 你可以进行任意次操作，每次操作为使某条边的边权增加 1。
- 求一种操作次数最小的方案使得从第一列任意一个点出发到最后一列任意一个点的最短路恰好增加了 K，输出方案。

E. Extending Distance

- 将网格图对偶，第 n 行以下的区域设为源点，第一行以上的区域设为汇点，原图中的任意一条路径对应了对偶图上的一个割，最短路就对应了对偶图上的最小割。
  (证明请参考 2006 ICPC Beijing Regional B: Animal Run)
- 我们将原图中的边费用设为 0，流量设为边权，再对每条边建费用为 1，流量为 $+\infty$ 的一条额外边，问题就转换成了找流量为 $D+K$ 的费用最小的流，其中 D 是原图的最短路。

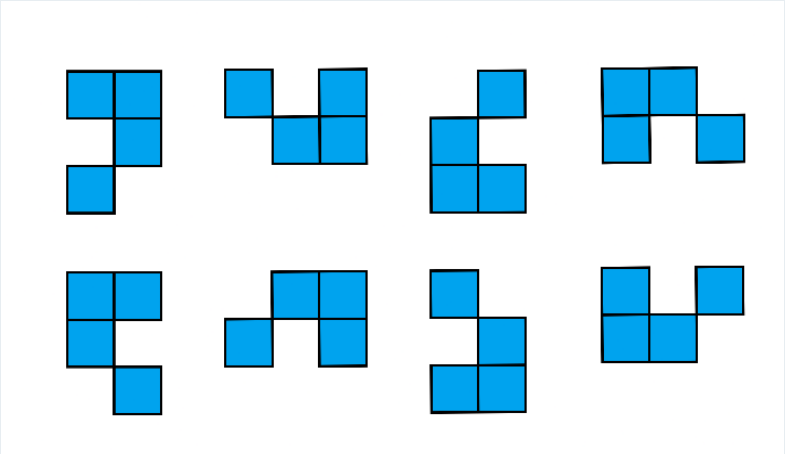
E. Extending Distance

- 原图的边权很大，而 K 很小，流量为 $D+K$ 的流中有 D 的流费用都为 0，我们可以先用 Dinic 跑最大流，把费用为 0 的流完，然后再拿普通费用流跑剩下的流。
- 输出方案只需要费用为 1 的所有边的反向弧的流量，就能知道每条边操作了多少次。
- 时间复杂度为 $\mathcal{O}(n^2m^2K)$。

H. Puzzle: Question mark

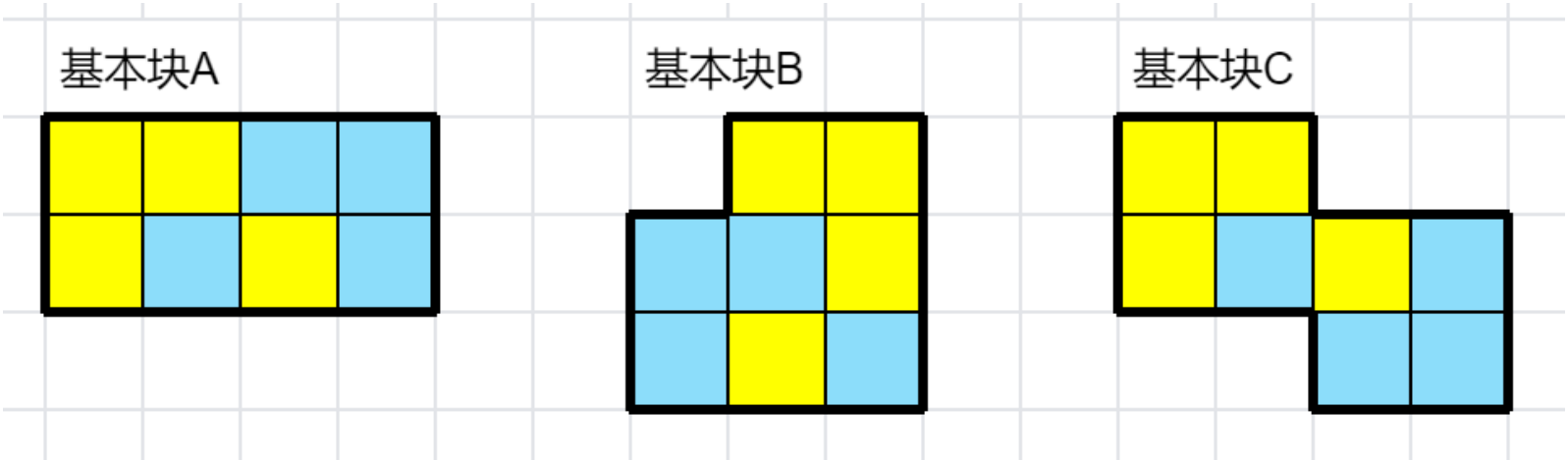
题意

- 在一个 $n \times n$ 的网格内不重叠地放入尽可能多的问号拼图（QM 拼图），QM 拼图可以旋转和翻面。
- 输出最多能放置的 QM 拼图数量以及对应的方案。



H. Puzzle: Question mark

单个 *QM* 拼图的形状过于不规则，我们可以将两块 *QM* 拼图以不同的方式拼起来，构成如下三种基本块（可旋转和翻转）：

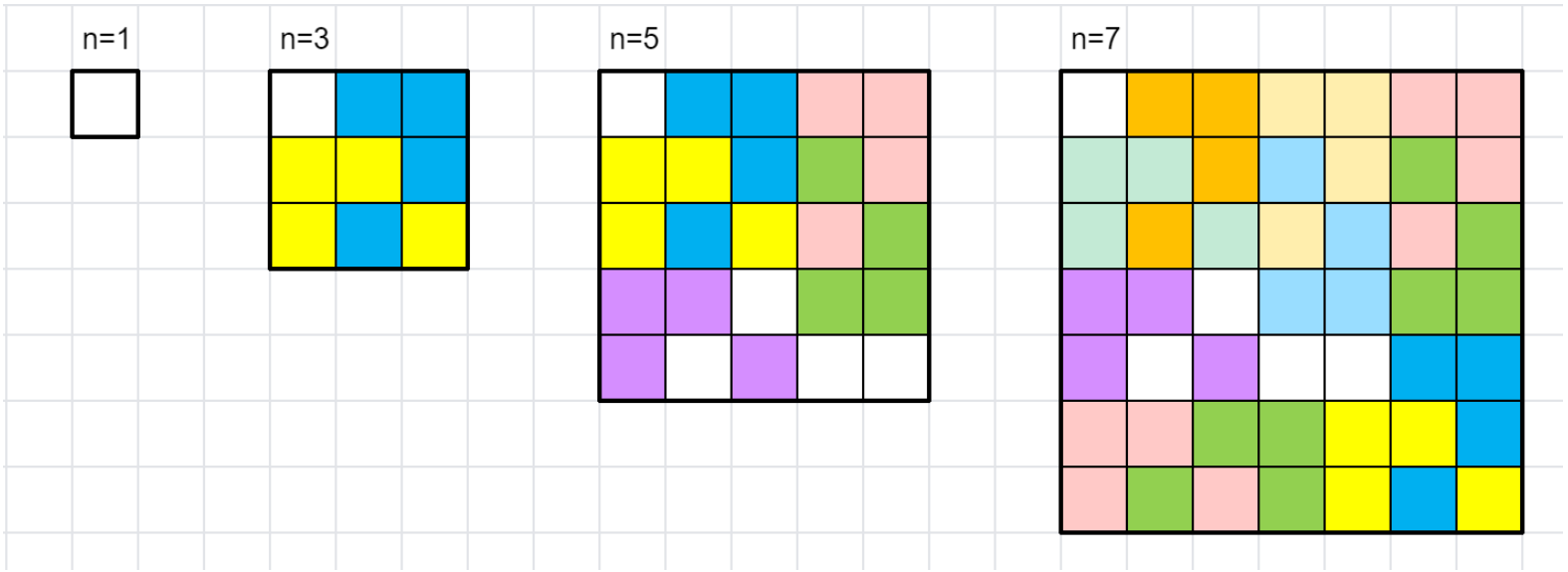


H. Puzzle: Question mark

- 对于 $n = 4k$ 的情况，只需从上至下，从左至右放置 $2k \times k$ 个 2×4 的基本块 A 即可恰好填满。
- 对于 $n = 4k + 2$ 的情况，首先不可能做到用 QM 拼图恰好填满，这是因为将网格黑白交替染色，黑白方格各有 $2 \times (2k+1) \times (2k+1)$ 个，QM 拼图需要 $(2k+1) \times (2k+1)$ 个，每个占据 3 黑 1 白或者 1 黑 3 白，最后占据的黑色格子数奇偶性不符。
- 接下来说明有 4 个格子空余的构造方案：先用 $(2k+1) \times k$ 个 2×4 的基本块 A 填满左侧 $(4k+2) \times 4k$ 的区域，再用 $k \times 1$ 个 4×2 的基本块 A 填满右上角 $4k \times 2$ 的区域，最后空出右下角 2×2 的区域。

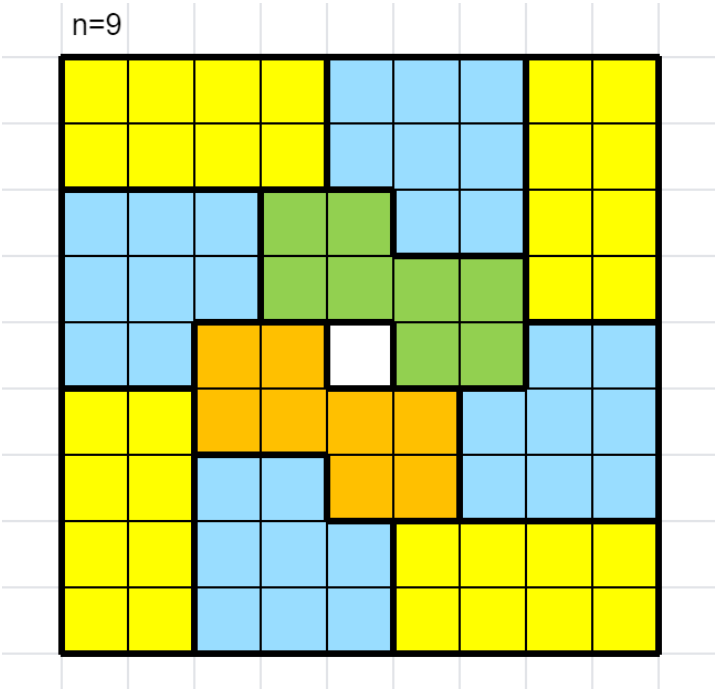
H. Puzzle: Question mark

然后是 n 为奇数的情况，下面是 $n = 1, 3, 5, 7$ 时的构造方案，并由爆搜程序证明该方案最优（$n = 5, 7$ 会空余 5 个）。



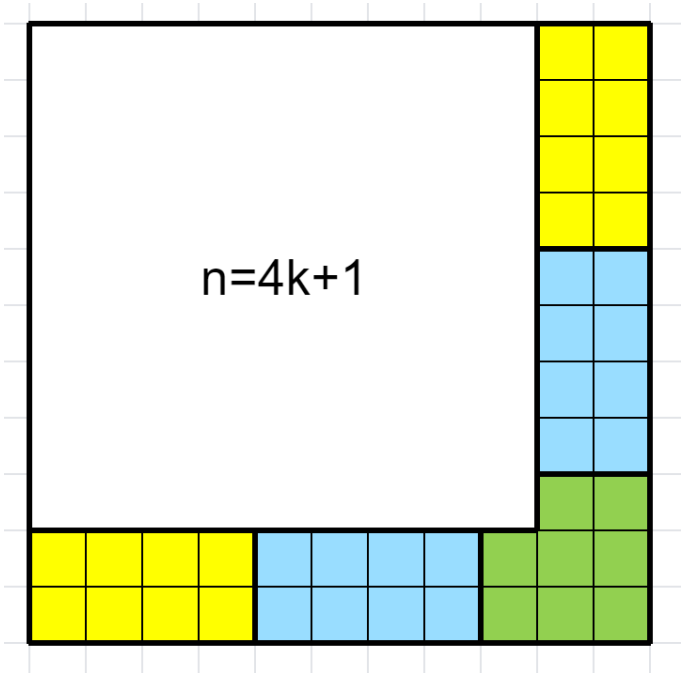
H. Puzzle: Question mark

- 对于 n 为奇数且 $n \geq 9$，均有构造方案使得只有 1 个空余格子（不同于 $n = 5, 7$ 会空余 5 个）
- $n = 9$ 的构造如下：



H. Puzzle: Question mark

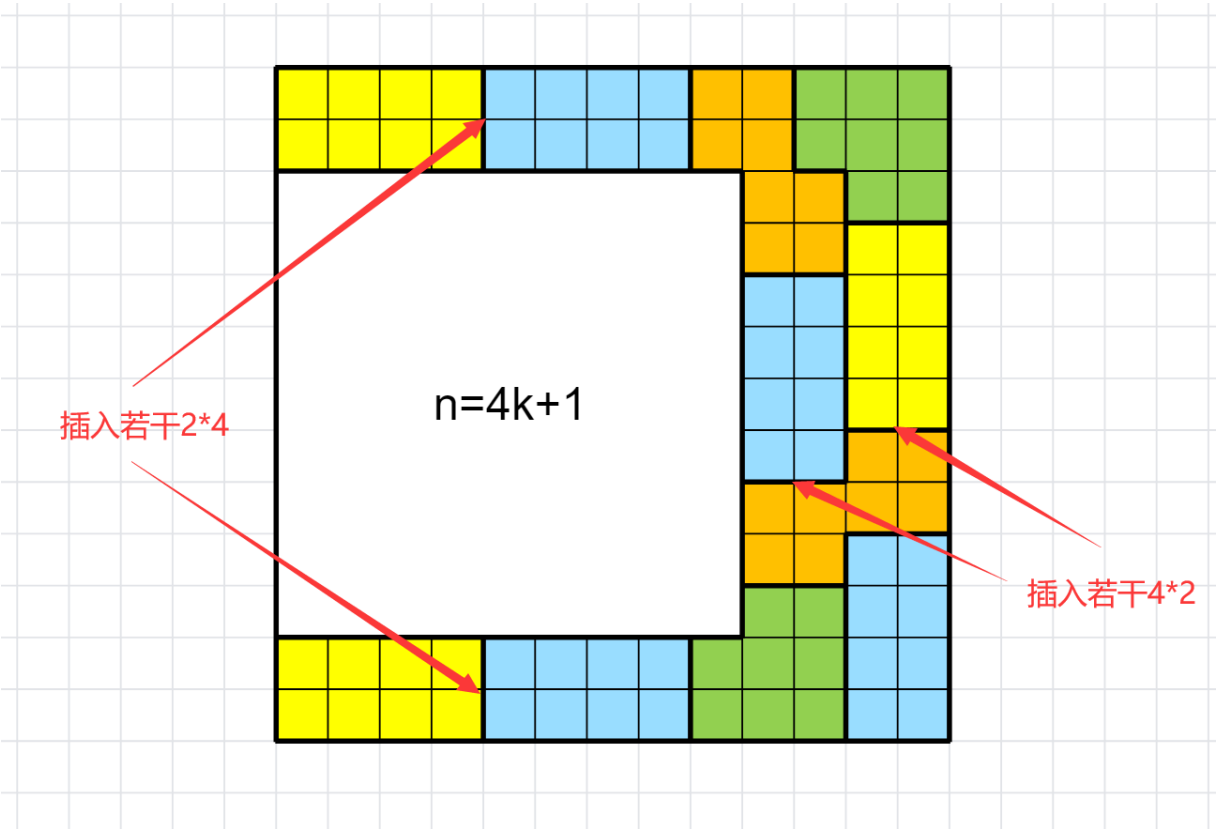
从 $n = 4k + 1$ 推出 $n = 4k + 3$ 的构造方案：



上图是 $9 \to 11$ 的例子，对于更大的情况只需在左下和右上添加对应数量的 2×4 和 4×2 的基本块 A 即可。

H. Puzzle: Question mark

从 $n = 4k + 1$ 推出 $n = 4k + 5$ 的构造方案：



上图是 $9 \to 13$ 的例子，对于更大的情况需要在上图箭头位置添加对应数量的 2×4 和 4×2 的基本块 A。

H. Puzzle: Question mark

- 总体来说，对于 $n \leq 8$ 或者 n 为偶数的情况，采取直接构造；对于 $n \geq 8$ 且 n 为奇数的情况，用上述两种递归构造最终归约到 $n = 9$ 的情况。
- 时间复杂度为 $O(n^2)$。
- 难点在于尽早发现 $n = 5, 7$ 不能几乎填满但 $n = 9$ 能几乎填满。

题意

- 给定长度为 m 的字符串 $t = t_1 \dots t_m$ 和一棵有 $(n+1)$ 个节点的树 T 每条边上都有一个字符。
- 考虑如下函数

$$f(i,j) = \max\{d(x) \mid s_x \text{ 是} s_i + \mathrm{pre}(t,j) \text{ 的后缀}\}$$

其中 s_i 是从根到节点 i 的最短路径上所有字符连接而成的字符串，$d(i)$ 是从根到节点 i 的最短路径经过的边数。

- 对每个 $1 \le j \le m$ 计算 $g_j = \sum_{i=1}^{n} f(i,j)$。

J. Suffix Structure

- 考虑每个 $f(i,j)$，有两种不同的情况。
- 第一种情况为 s_x 为 $\mathrm{pre}(t,j)$ 的后缀。那么，我们在 trie 树上建立 AC 自动机，并将 t 在 AC 自动机上匹配。匹配到节点在 fail 树上的所有祖先均为 $\mathrm{pre}(t,j)$ 的后缀。
- 由于字符集的大小较大，所以需要使用可持久化线段树进行维护。
- 否则，相当于 s_x 为一个 s_i 的一个后缀拼接上 t。也就是说，如果我们从 i 对应的节点往下去匹配 $t[1:j]$ 走到一个节点 p，满足 p 跳若干次 fail 后能够跳到 x。
- 在这种情况下，$|s_x| > j$，因此我们一定形如从树上的某个点 u 开始往后走，往下走 j 步走到 x。
- 我们考虑去找到所有可行的 u，即从 u 这个点开始往下走，能够匹配的长度是 $\geq j$ 的。

J. Suffix Structure

- 我们对串 t 运行 Z-algorithm，对每个 $i = 1, 2, \cdots, |t|$ 计算出 $Z_i = |\mathrm{lcp}(t[i, |t|], t)|$。
- 对每一个点 v，假设它的祖先里能匹配到 v 的自述里的点匹配最深的一个点是 u（并记其匹配到 v 子树内深度为 D 的地方），那么 v 的匹配长度可以被 $\min(D - d_v, f[d_v - d_u + 1])$ 更新。
- 每个点 v 只需要将其本身以及祖先们往其子树内匹配最深的串 s' 传到相应的子树内，其余子树的信息可以忽略。我们可以进行暴力更新匹配，均摊 match 的复杂度可以保证。

J. Suffix Structure

- 在求出了树上每个点 u 跟 s 串匹配的最大长度 L_u 后，对于一个 j，所有可能贡献某个 x 的 u 是满足 $L_u \geq j$ 的所有 u。
- 倒着枚举 j，将可行的 u 一个一个加入到 fail 树中。
- 考虑 i 对 s_j 前缀跳 fail 的贡献，问题可以转化成以下形式：

问题

- 给定一棵树，每次标记一个未标记过的点，或者询问树中所有点到他最近的被标记的祖先的距离之和。

- 对于转换后的问题，每次标记一个点的操作相当于把这个点的子树 cut 下来。
- 可以使用线段树分裂和合并来进行维护。
-

B. Intersection over Union

题意

- 给定一个矩形（OBB），要求找到一个坐标轴平行的矩形（AABB）。
- 这两个矩形的面积交除以面积并叫做交并比（IOU）。
- 求 IOU 的最大值。

B. Intersection over Union

这是一个函数求极值问题，一般会考虑

- 是否单调/单峰/凸？=> 二分/三分
- 是否可导/可微？=> 解方程（组）/梯度下降/牛顿迭代
- 局部最小陷阱？=> 爬山/模拟退火
- 有没有约束？=> 拉格朗日乘子法/KKT/线性规划...

B. Intersection over Union

- 利用对称性可以简化问题。答案显然只和 OBB 的长宽比以及角度有关，我们对 OBB 做平移，镜像都不影响答案。
- 不妨把这个 OBB 平移，使得它的中心位于原点。
- AABB 参数化为 (x, y, w, h)，其中心位于 (x, y)，上下左右边界分别为 $y+\frac{h}{2}$, $y-\frac{h}{2}$, $x-\frac{w}{2}$, $x+\frac{w}{2}$.

$$IOU(x, y, w, h) = \frac{I_{w,h}(x, y)}{S_{OBB} + wh - I_{w,h}(x, y)}$$

- $I_{w,h}(x, y)$ 是 AABB 与 OBB 交的面积，根据中心对称，$I_{w,h}(x, y) = I_{w,h}(-x, -y)$ 是个偶函数。
- $I_{w,h}(x, y)$ 同时也是凸函数。(凸性可以参考 QOJ 7350)

$$I_{w,h}(0, 0) = I_{w,h}(\frac{x-x}{2}, \frac{y-y}{2}) \geq \frac{I_{w,h}(x, y) + I_{w,h}(-x, y)}{2} = I_{w,h}(x, y)$$

- 所以 $IOU(0, 0, w, h) \geq IOU(x, y, w, h)$ ，AABB 的中心放在原点，也就是与 OBB 的中心重合是最优的。

B. Intersection over Union

- 设 $g(w,h) := IOU(0,0,w,h)$，$u(h)$ 是面积并关于 h 的导数，那 $g(w,h)$ 可看成关于 h 的函数表示如下:

$$g(w,h) = \frac{union + intersection}{union} - 1 = \frac{S_{OBB} + w \times h}{S_{OBB} + \int_0^h u(t)dt} - 1$$

$$\frac{\partial g}{\partial h} = \frac{w\left(S_{OBB} + \int_0^h u(t)dt\right) - (S_{OBB} + wh)u(h)}{\left(S_{OBB} + \int_0^h u(t)dt\right)^2}$$

$$= \frac{S_{OBB} \times (w - u(h)) + w\int_0^h (u(t) - u(h))\,dt}{\left(S_{OBB} + \int_0^h u(t)dt\right)^2}$$

- 注意到 $u(h)$ 是关于 h 单调不递减且满足 $0 \le u(h) \le w$, 这里的分子也是单调不递增的，而分母恒大于 0，故 g 关于 h 是单峰的。对称地，g 关于 w 也是单峰的。
- $g(w,h)$ 整体是单峰函数。可以利用三分法，爬山法，梯度下降法等求解，都可以收敛到全局最优解。接下来介绍公式方法。

B. Intersection over Union

- 为了方便讨论，可以通过镜像以及等比缩放等操作（不影响答案），使得 OBB 长边为 $\cos\phi$，短边 $\sin\phi$，长边和 x-轴的夹角为 θ。$\left(0 < \phi \le \frac{\pi}{4}, 0 \le \theta \le \frac{\pi}{4}\right)$
- 进行分类讨论，最优解时，交集可能是矩形（$\theta = 0$），六边形（OBB 细长且倾斜度较大时）和八边形。我们也可以对每种情况列出函数，然后求解梯度等于 0 的方程来获得公式解。

$$\max g(w,h) = \begin{cases} 1, & \text{if } \theta = 0 \\ \frac{2}{\sqrt{1+8\cot\phi\sin 2\theta}-1}, & \text{if } \cot\phi\sin 2\theta \ge 3 \\ \frac{k}{\sin 2\theta - k}, & \text{otherwise} \end{cases}$$

$$k = -\frac{\csc 2\phi}{3} + \frac{2p}{3}\cos\left(\frac{1}{3}\arccos\left(-\frac{q}{2p^3}\right) - \frac{2\pi}{3}\right)$$

$$p = \sqrt{\csc^2 2\phi + 3\sin 2\theta \csc 2\phi + 6}$$

$$q = 2\csc^3 2\phi + 9(\csc^2 2\phi + 3)\sin 2\theta + 18\csc^2 2\phi$$

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!