IOI2015 中国国家集训队第一次作业
试题泛做表格

江苏省常州高级中学 张志俊

目录

Chapter 1

Codeforces

1.1 235C-Cyclical Quest (D9164,P6045)

题目大意

给定母串为 s，给出 n 组询问 X_i，每次询问有多少 s 的子串循环同构于模式串 X_i。

数据范围

- 定义 $sum(length(X_i))$ 表示字符串 X_i 的总长度。
- 对于 30% 的数据，$length(s) \leq 500$，$n \cdot sum(length(X_i)) \leq 500$；
- 对于 50% 的数据，$length(s) \leq 500$，$n \leq 300$，$sum(length(X_i)) \leq 1500$；
- 对于 100% 的数据，$length(s) \leq 10^6$，$n \leq 10^5$，$sum(length(X_i)) \leq 10^6$。

算法分析

本题由于给出的字符串总长度过大，因此只能借助后缀自动机来解决。不难发现，本题就是一道后缀自动机的模板题，每次询问时只需要将 X_i 复制一遍，然后在自动机上直接进行询问即可。整个算法的时间复杂度为 $\Theta(|S||s| + \sum |X_i|)$，其中 S 表示字符集。

时间复杂度

$\Theta(|S||s| + \sum |X_i|)$

空间复杂度

$\Theta(|S||s|)$

1.2 235D-Graph Game (D9293,P6104)

题目大意

给定一张 n 个点 n 条边的环套树，执行随机点分治，定义单次分治的代价为当前整棵子树中的点数。求随机点分治的期望总代价。

数据范围

- $0 < n \leq 3000$。

算法分析

对于本题，我们不妨首先考虑树上的情形，则任意两点之间仅有一条唯一的路径。由于期望的线性性，则任意点对 (x, y) 对于期望的贡献为，当点分治进行到点 x 时，点 y 依然与点 x 连通的概率。易知，这一概率显然等于 $\frac{1}{len}$，其中 len 表示树上 x 到 y 之间的距离。

下面考虑环套树的情形，若 x 和 y 处于同一棵子树中，则贡献必然不变，否则，我们只需求得两棵子树对应的根在环上的距离，然后进行一次容斥即可。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n)$

1.3 235E-Number Challenge (D8809,P5483)

题目大意

给定 a, b, c，求 $\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(ijk)$，其中 $d(x)$ 表示 x 约数的个数。

数据范围

- $1 \leq a, b, c \leq 2000$。

算法分析

通过一定的数学推导可以得到以下等式（对于任意多维均成立，详细证明参见这里）：

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(ijk) = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} \lfloor\frac{a}{i}\rfloor\lfloor\frac{b}{j}\rfloor\lfloor\frac{c}{k}\rfloor[gcd(i,j) = gcd(j,k) = gcd(i,k) = 1]$$

在得到上述结论后，不难借助莫比乌斯反演来降低复杂度。另外，最大公约数可以在预处理中通过递推直接得出。整个算法的时间复杂度为 $\Theta(n^2 \log n)$，其中 $n = max(a, b, c)$。

时间复杂度

$\Theta(n^2 \log n)$

空间复杂度

$\Theta(n^2)$

1.4 238D-Tape Programming (D8807,P5488)

题目大意

对于一个给定的字符串，规定初始位置为最左侧，初始方向为向右。对于每一步操作：

- 若当前位置的字符是数字，则输出一次该数字。同时，若该数字是 0，则删除该位置的字符，然后按当前方向前进一步；若该数字不是 0，则将该位数字减一，然后按当前方向前进一步；
- 若当前位置的字符为'<' 或者'>'，则将当前的方向对应地修改为向左或者向右，然后按新的方向前进一步。若新位置的字符仍然不是数字，则删除之前位置的字符。

若在该过程中的任意时刻，当前位置已经不处于字符串内，则操作结束。

现在给定一个长度为 n 的字符串 s，然后进行 q 次询问，每次查询：若将字符串 s 的第 l_i 到 r_i 位单独作为一个字符串，则在对该字符串操作的整个过程中，每种数字分别被输出了多少次。

数据范围

- 对于 30% 的数据，$n \leq 100$，$q \leq 100$；
- 对于 100% 的数据，$n \leq 10^5$，$q \leq 10^5$，$1 \leq li \leq ri \leq n$。

算法分析

首先，由于初始方向为向右，且初始位置在最左侧，所以在对整个字符串 s 进行操作的过程中，当前位置首次到达某一位置 i 时，此时的方向也一定是向右。因此，题中若干次询问中的操作序列，其实都是对整个字符串 s 的操作序列中的一段连续子串。

于是，每一次的询问实质上就是先找到 l_i 首次在完整操作序列中出现的位置，然后再找到 $l_i - 1$ 以及 $r_i + 1$ 在该位置之后首次出现的位置。因此，我们可以先对整个字符串 s 模拟其操作过程，并记下各种数字出现次数的前缀和，然后每次询问只需二分位置即可。

在预处理中，由于每次操作要么使得某个数字减一，要么删掉某个非数字的字符，因此总的操作次数不超过 $11n$。整个算法的时间复杂度为 $\Theta(n + q \log n)$。

时间复杂度

$\Theta(n + q \log n)$

空间复杂度

$\Theta(n)$

1.5 240F-Torcoder (D9159,P6050)

题目大意

给定一个长度为 N 的字符串，给出 M 组询问。对于每次询问 $[l,r]$，要求对位于 $[l,r]$ 内的所有字符重新排列，使得重新排列后的子串变成可能的字典序最小的回文串。若某次询问存在合法解，则执行该次操作，否则不发生任何改变。求所有 M 次询问全部执行完毕之后的字符串。

数据范围

- 对于 40% 的数据，$N \leq 1000$，$M \leq 1000$；
- 对于 100% 的数据，$N \leq 100000$，$M \leq 100000$。

算法分析

显然，若已知给定区间中每种字符出现的总次数，则必然可以直接贪心地重新排列整个序列。因此，我们可以直接借助线段树来维护每种字符出现的总次数，所需用到的操作仅有区间赋值。整个算法的时间复杂度为 $\Theta(|S|^2 m \log n)$。

时间复杂度

$\Theta(|S|^2 m \log n)$

空间复杂度

$\Theta(|S|n)$

1.6 241D-Numbers (D9233,P5531)

题目大意

给定一个 $1 \sim n$ 的排列，要求从中选取出一个子序列，必须满足：

- 子序列非空；
- 子序列中所有整数的异或和为 0；
- 将子序列中的所有整数均看做字符串，并依次进行拼接，使得形成的大整数在十进制表示下被 p 整除。

要求找出任意一组满足上述所有条件的可行方案。

数据范围

- 对于 10% 的数据，$n \leq 18$；
- 对于 35% 的数据，$n \leq 30$；
- 对于另外 15% 的数据，$n \leq 100$，$p \leq 100$；
- 对于 100% 的数据，$1 \leq n, p \leq 50000$，且 p 是质数。

算法分析

对于本题，朴素 DP 的总代价为 $\Theta(n^2p)$，显然根本无法承受。但是，由于给定的序列是 $1 \sim n$ 的一个排列，则 $1 \sim n$ 以内的每个正整数均会恰好出现一次，因此不妨近似地将序列中的数字看做具有一定的随机性。

接下来，我们考虑仅使用 2^l 以内的所有正整数，则根据归纳证明显然可以发现，任意选择其中若干个正整数所得到的异或和完全是平均分布的，因此，异或和为 0 的方案共有 2^{2^l-1-l} 种。其次，当 p 与 10 互质时，拼接出的大整数模 p 的结果近似也是随机的，即模 p 为 0 的方案数大约有 $\frac{2^{2^l-1}}{p}$ 种。同时，由于两种限制几乎完全不相关，因此若能够选取到恰当的 l，则有极大的概率能够较快地找到合法解。在本题中，通过实践可以发现，我们只需选取 $l = 5$ 即可。整个算法的时间复杂度为 $\Theta(2^{2l}p)$。

时间复杂度

$\Theta(2^{2l}p)$

空间复杂度

$\Theta(2^{2l}p)$

1.7 241E-Flights (D9227,P5698)

题目大意

给定一张 n 个点 m 条边的有向图，初始时所有边权均为 1，要求将其中若干条边的权值修改为 2，使得任意一条从 1 到 n 的路径长度全部相等。求任意一组合法方案。

数据范围

- 对于 20% 的数据，$2 \leq n \leq 10$，$1 \leq m \leq 20$；
- 对于 60% 的数据，$2 \leq n \leq 100$，$1 \leq m \leq 300$；
- 对于 100% 的数据，$2 \leq n \leq 1000$，$1 \leq m \leq 5000$。

算法分析

首先，易知所有不能从 1 到达或者不能到达 n 的点对于最终方案均没有任何影响，因此可以先排除所有这类点。对于剩下的所有点，由于从 1 到 n 的所有路径要求全部等长，因此从 1 到任意一个点 i 的距离 d_i 必然是唯一的。于是，我们可以直接根据每条边的限制构建出差分约束系统，然后只需执行最短路算法即可解决。整个算法的时间复杂度为 $\Theta(nm)$。

时间复杂度

$\Theta(nm)$

空间复杂度

$\Theta(n+m)$

1.8 241F-Race (D9204,P5847)

题目大意

给定一张 $n \times m$ 的街区地图，地图由若干建筑物、双向的直线道路和交叉路口组成。数据保证，每栋建筑物恰好占一个街区，所有道路的宽度都为一个街区，并且所有道路均为水平方向或者竖直方向的，每个交叉路口也恰好占一个街区，均位于道路两两相交的位置上。定义两个街区相邻当且仅当两个街区之间存在公共边，数据保证任意两条道路或两个交叉路口不相邻。

现在给定起点所在的街区，保证起点一定位于某条道路上。给定经过所有交叉路口的顺序，要求依次经过这些交叉路口，且完成任务后停留在原地。同时，在交叉路口需要额外花费 1 单位的时间来到达相邻街区。另外，要求行进途中不能经过任何建筑物。求出发 k 单位时间之后所在的位置。

数据范围

- 对于 100% 的数据，$3 \leq m, n \leq 100$，$1 \leq k \leq 100000$。

算法分析

本题主要是题面比较难以理解，在搞清了若干细节后只要直接模拟就行了。整个算法的时间复杂度为 $\Theta(nm + k)$。

时间复杂度

$\Theta(nm + k)$

空间复杂度

$\Theta(nm)$

1.9 243C-Colorado Potato Beetle (D8745,P5652)

题目大意

给定一张 $(10^{10}+1)\times(10^{10}+1)$ 的网格，初始时处于网格的中心处。给出 n 次操作，每次操作为沿着某个给定方向（上、下、左、右）行进一段给定的距离 x_i。求在所有操作结束后，被行进轨迹围出的总面积有多大。

数据范围

- $1\leq n\leq 1000$，$1\leq x_i\leq 1000000$。

算法分析

对于这一问题，虽然给定的区域面积已经超出了长整型的范围，但是稍加分析即可发现，由于每步的移动距离最多只有 10^6，因此所有操作的活动范围也不会超过 10^9，因此答案的大小仍然可以承受。

其次，由于本题的操作次数远小于活动范围，因此不难想到预先将所有涉及到的横纵坐标分别离散化，则离散化后的一个网格代表了离散化前的一块矩形区域，且该块矩形区域内所有网格的最终状态应该是一致的。因此，只需在离散化后暴力地标记出行进轨迹，然后从整个区域的边界开始 $flood-fill$ 即可，整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n^2)$

1.10 243D-Cubes (D8766,P5583)

题目大意

给定平面直角坐标系，在 $(0,0)$ 到 (n,n) 之间的每个单位网格内堆积已知高度的单位立方体。给定方向向量 $\vec{v}=(v_x,v_y)$，求从该方向水平观察可以看到的单位立方体个数。

数据范围

- $1 \le n \le 10^3$，$0 \le a_{i,j} \le 10^9$；
- $|v_x|,|v_y| \le 10^4$，$|v_x|+|v_y|>0$。

算法分析

首先易知，每个单位网格上可见的单位立方体均是自顶部开始向下的连续一段，因此只需对每个单位网格求得其最低可见的单位立方体高度即可。其次，某个单位立方体可见必然完全等价于：在俯视图中，该单位网格内存在某一点，使得以该点为端点、方向为 $-\vec{v}$ 的射线经过的所有网格内立方体高度均低于该立方体。

在该等价条件中，由于只要存在这样的一点即可，因此可以转化为查询一段区间内的最低高度。因此，考虑旋转坐标系，使得方向 $-\vec{v}$ 成为坐标轴，则在确定所有单位网格的操作顺序以后，只剩下两类操作：

- 查询一段区间内的最小值；
- 将一段区间内的所有值与给定值取 max。

显然，这两个操作只需借助线段树即可解决。整个算法的时间复杂度为 $\Theta(n^2 \log n)$。

时间复杂度

$\Theta(n^2 \log n)$

空间复杂度

$\Theta(n^2 \log n)$

1.11 243E-Matrix (D8748,P5631)

题目大意

给定一个 $n \times n$ 的 01 矩阵，判断是否可能通过重新排列某些列的顺序，使得每一行的所有 1 都连续，求任意一组合法方案。

数据范围

- 对于 20% 的数据，$n \leq 9$；
- 对于 40% 的数据，$n \leq 24$；
- 对于 100% 的数据，$n \leq 500$。

算法分析

本题为一道 $PQ-tree$ 的模板题，只需要根据每一行的限制依次对树进行维护即可，单次操作的代价为 $\Theta(n)$。在一棵 $PQ-tree$ 中，所有的 P 类节点表示其子节点可以为任意顺序，所有的 Q 类节点则表示其子节点只能为当前顺序或者当前顺序的逆序。在维护过程中，我们只需自底向上对每个涉及的节点分别对应到 9 大类模板中即可。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n^2)$

1.12 248E-Piglet's Birthday (D8744,P5663)

题目大意

给定 n 个架子，已知每个架子上的初始蜜罐个数为 a_i，且初始时所有蜜罐均为满的。共进行 q 次操作，每次从第 u_i 个架子上随机取出 k_i 个蜜罐，将取出的蜜罐中剩余的蜜全部吃完，然后将这 k_i 个空蜜罐全部放在第 v_i 个架子上。求每次操作之后，期望有多少个架子上的蜜罐全部为空。

数据范围

- $1 \le n \le 10^5$，$1 \le q \le 10^5$；
- $0 \le a_i \le 100$，$1 \le u_i, v_i \le n$，$1 \le k_i \le 5$。

算法分析

由于题目中要求的是蜜罐全部为空的架子的期望个数，且每个架子上满的蜜罐个数较少，因此我们不妨维护在每步操作之后，每个架子上满的蜜罐个数恰好为某一值的概率，则统计答案时只需将所有架子上满的蜜罐个数为 0 的概率累加即可。

- 若记 $pr[i][j]$ 为在当前一步操作结束之后，第 i 个架子上满的蜜罐个数恰好为 j 的概率，则单次操作 u_x v_x k_x 只会涉及 $pr[u_x]$；
- 由于每次操作取 k_x 个蜜罐较为繁琐，不妨改为每次操作只取一个蜜罐，则给出的每步操作可被等价地分解为 k_x 次操作；
- 由于每步操作只取一个蜜罐，则被取出的蜜罐只有满和不满两种情形，因此若记 $tot[i]$ 为当前第 i 个架子上蜜罐的总数，则不难得出如下 DP 方程：

$$pr[i][j] = pr[i][j] \times \frac{tot[i]-j}{tot[i]} + pr[i][j+1] \times \frac{j+1}{tot[i]}$$

- 其中，前半部分为取出的蜜罐为空的情形，后半部分为取出一个满的蜜罐的情形。

借助上述的 DP 方程，则每次操作结束之后的询问只需累加所有 $pr[i][0]$ 即可，整个算法的时间复杂度为 $\Theta(aq)$。

时间复杂度

$\Theta(aq)$

空间复杂度

$\Theta(an)$

1.13 249D-Donkey and Stars (D8770,P5557)

题目大意

给定平面直角坐标系中的 n 个已知点，每个点只能转移到仰角在 α_1 到 α_2 之间的点，求一个最长的点序列，使得相邻的两点之间均可以转移。

数据范围

- 对于 20% 的数据，$n \leq 15$；
- 对于 100% 的数据，$1 \leq n \leq 10^5$，$0 \leq a, b, c, d \leq 10^5$，$0° \leq \alpha_1 < \alpha_2 \leq 90°$，$a + b > 0$ 且 $c + d > 0$，$0 \leq x, y \leq 10^5$。

算法分析

易知，由于给定的可视范围是一个仰角区间，因此不难通过简单的向量运算将 α_1 和 α_2 转化为两条坐标轴，则每个点的可见点就是相对于它处于第二象限内的所有点。对于这一经典问题，我们只需要在降维后将其看做最长上升子序列问题即可。整个算法的时间复杂度为 $\Theta(n \log n)$。

时间复杂度

$\Theta(n \log n)$

空间复杂度

$\Theta(n)$

1.14 251D-Two Sets (D9208,P5769)

题目大意

给定 n 个非负整数 a_i，要求将所有数分入两个集合，使得两个集合内所有元素异或和的总和最大，同时要求在此前提之下最小化第一个集合中所有数的异或和。

数据范围

- 对于 30% 的数据，保证 $n \leq 10$；
- 对于 60% 的数据，保证 $n \leq 1000$；
- 对于 100% 的数据，保证 $n \leq 100000$。

算法分析

首先考虑所有数的异或和，若其二进制表示下的某一位为 1，则显然无论如何分割，两个集合中对应位置必然分别是 0 和 1；若其二进制表示下的某一位为 0，则两个集合中的对应位置可能同时为 0 或者同时为 1，显然要使总贡献最大，则必然应该尽可能组合出 1。通过分析可以发现，若我们从高位向低位依次处理，则只需借助高斯消元即可解决。整个算法的时间复杂度为 $\Theta(n \log^2 a)$。

时间复杂度

$\Theta(n \log^2 a)$

空间复杂度

$\Theta(n \log a)$

1.15 254D-Rats (D8765,P5584)

题目大意

给定一张 $n \times m$ 的网格，其中部分网格为障碍格，且已知某些非障碍格中有老鼠，其余网格均为空。要求找到一种合法方案，在两个不同的非障碍格中分别放置一枚炸弹，使得所有老鼠均被消灭。对于每一枚炸弹，所有与其曼哈顿距离不超过 d 的网格中的老鼠均会被消灭。

数据范围

- 对于 30% 的数据，$1 \leq n, m \leq 10$；
- 对于 100% 的数据，$1 \leq n, m \leq 1000$，$1 \leq d \leq 8$。

算法分析

由于要求清除所有的老鼠，因此可以首先任选一只老鼠，则必然至少有一枚炸弹与其曼哈顿距离不超过 d，易知满足条件的网格至多只有 $\Theta(d^2)$ 个。考虑枚举第一枚炸弹的位置，则剩余的所有老鼠必然只能被第二枚炸弹消灭。类似的，我们从剩余的老鼠中同样任选出一只，则第二枚炸弹可能的位置也只有 $\Theta(d^2)$ 个，只需再遍历一次即可验证方案的可行性。整个算法的时间复杂度为 $\Theta(nm + d^6)$。

时间复杂度

$\Theta(nm + d^6)$

空间复杂度

$\Theta(nm)$

1.16 258D-Little Elephant and Broken Sorting (D9195,P5911)

题目大意

给定一个 $1 \sim n$ 的排列 p，依次进行 m 次交换操作，每次操作将交换第 a_i 个和第 b_i 个位置上的整数，但每次操作均只有一半的概率执行，一半的概率不执行。求执行完所有操作后，最终排列中逆序对个数的期望值。

数据范围

- $1 \leq n, m \leq 1000$，$n > 1$；
- $1 \leq a_i, b_i \leq n$，$a_i \neq b_i$。

算法分析

对于这一问题，由于要求期望的逆序对个数，因此考虑维护第 i 个值大于第 j 个值的概率 $dp[i][j]$，则对于每次操作而言，造成的影响只有所有涉及到位置 i 或者位置 j 的概率值，因此单次维护的代价为 $\Theta(n)$。整个算法的时间复杂度为 $\Theta(nm)$。

时间复杂度

$\Theta(nm)$

空间复杂度

$\Theta(n^2)$

1.17 260E-Dividing Kingdom (D8743,P5664)

题目大意

在平面直角坐标系中给定 n 个整点，并给定序列 $a_1, a_2, \ldots, a_9$。询问是否可能找到两条不重合的水平直线以及两条不重合的竖直直线，使得平面上被分成的 9 个区域中出现的点数，是序列 $\{a\}$ 的某个排列。要求给出具体分割方案。

数据范围

- $9 \leq n \leq 10^5$，$-10^9 \leq x, y \leq 10^9$；
- $1 \leq a_i \leq 10^5$，$a_1 + a_2 + \ldots + a_9 = n$。

算法分析

由于题中并未要求每一块区域中出现的点数恰好为多少，而只要满足是序列 $\{a\}$ 的某个排列，因此必须首先确定每一块区域中的具体点数，然后才能进行验证，其中总的枚举量为 9!。于是，问题转化为已知每一块区域内确切的点数，判断是否存在合法的分割方案。

对于这一问题，首先根据纵向的三块区域内点数的总和，我们可以方便地通过二分法确定两条纵向直线的范围。其次，对于纵向的三块区域而言，我们需要分别求出满足点数要求的水平直线范围，然后分别合并三个可行区间即可。而在确定水平直线的范围时，所有的询问均可转化为查询前 i 行前 j 列中的总点数，因此只需在其中一维坐标上建立可持久化线段树即可解决。整个算法的时间复杂度为 $\Theta(n \log n + 9! \cdot \log n)$。

时间复杂度

$\Theta(n \log n + 9! \cdot \log n)$

空间复杂度

$\Theta(n \log n)$

1.18 261D-Maxim and Increasing Subsequence (D9279,P5606)

题目大意

给定一个长度为 n 的初始序列 a，已知其中的每个元素均不超过 $maxb$。要求将 a 复制 t 次以构成一个新序列。求构成的新序列的最长上升子序列。

数据范围

- 对于 30% 的数据，$n \cdot t \leq 10^5$；
- 对于 100% 的数据，$1 \leq k \leq 10$，$1 \leq n, maxb \leq 10^5$，$1 \leq t \leq 10^9$，$n \cdot maxb \leq 2 \times 10^7$。

算法分析

对于本题，原始的做法单次转移的代价为 $\Theta(\log maxb)$，因此必须考虑进行优化。在朴素做法中，我们记录了每种长度的上升子序列末位元素的最小值，每次转移都要在其中进行二分。不过考虑到本题的特殊性，我们考虑记录 $f[i]$ 为到当前位置为止，所有以不超过 i 的元素作为结尾的上升子序列的最大长度，则 f 的单调性显然。对于每次询问，则我们需要修改所有的 $f[j]$（$i < j$ 且 $f[i] = f[j]$）。同时，由于 f 的值显然不可能超过 n，则总的修改量不超过 $n \cdot maxb$。另外，由于 $t > maxb$ 的部分没有意义，因此总代价可以承受。整个算法的时间复杂度为 $\Theta(kn \cdot maxb + knt)$。

时间复杂度

$\Theta(kn \cdot maxb + knt)$

空间复杂度

$\Theta(n + maxb)$

1.19 261E-Maxim and Calculator (D8791,P5514)

题目大意

已知一个二元组 $(1,0)$，每步操作可以将二元组 (a,b) 变为 $(a,b+1)$ 或者 (ab,b)。求在区间 $[l,r]$ 内有多少个数可以通过不超过 p 步操作变为二元组的第一个数。

数据范围

- 30% 的数据满足，$2 \le l \le r \le 10^6$，$1 \le p \le 15$；
- 另外 20% 的数据满足，$2 \le l = r \le 10^9$，$1 \le p \le 100$；
- 100% 的数据满足，$2 \le l \le r \le 10^9$，$1 \le p \le 100$。

算法分析

首先，由于二元组的第二个数每次只能加一，因此第一个数的质因子均不超过 p，所以我们可以预先筛出所有质因子不超过 p 的正整数。其次，我们可以枚举最后状态中二元组第二个数的取值，然后直接 DP 即可。整个算法的时间复杂度为 $\Theta(\pi(p)r)$。

时间复杂度

$\Theta(\pi(p)r)$

空间复杂度

$\Theta(r)$

1.20 263E-Rhombus (D8738,P5693)

题目大意

给定一张 $n \times m$ 的网格表，每个网格内给定一个非负整数。要求在横坐标属于 $[k, n-k+1]$ 并且纵坐标属于 $[k, m-k+1]$ 的网格中选取一个网格，使得其对应的函数值最大。其中，给定的函数 $f(x,y)$ 定义为：对于所有与 (x,y) 的曼哈顿距离不超过 k 的网格 (x',y')，将 $a_{x',y'} \cdot (k - |x-x'| - |y-y'|)$ 求和。

数据范围

- 对于 25% 的数据，$1 \le n, m \le 100$；
- 对于 100% 的数据，$1 \le n, m \le 1000$，$1 \le k \le \lfloor \frac{min(n,m)+1}{2} \rfloor$，$0 \le a_{i,j} \le 1000000$。

算法分析

首先不难发现，与 (x,y) 曼哈顿距离不超过 k 的网格构成一个正方形，并且左上、右上、左下、右下四个部分完全对称，因此只需讨论其中任意一种情形即可，最后将四个部分全部相加就能得到每个网格的函数值。

接下来我们仅考虑左上方的三角形部分。考虑从 (i,j) 转移到 $(i,j+1)$，我们可以将此过程进行如下的转化：

- 首先在以 (i,j) 为顶点的 k 阶三角的右侧附加上长度为 $k+1$ 的一列，形成以 $(i,j+1)$ 为顶点的 $k+1$ 阶三角；
- 然后除去斜边上的 $k+1$ 个元素，形成以 $(i,j+1)$ 为顶点的 k 阶三角。

通过这样的转化，所需用到的所有值便都可以通过预处理得到。整个算法的时间复杂度为 $\Theta(nm)$。

时间复杂度

$\Theta(nm)$

空间复杂度

$\Theta(nm)$

1.21 264D-Colorful Stones (D8793,P5512)

题目大意

给定两个长度分别为 n 和 m 的 RGB 序列，已知两个人初始时分别站在两个序列的开头位置。每步操作只能是 RGB 之一，则位于相应颜色上的人会向右移动一位。要求任意时刻两个人均不能位于序列之外，求可能从初始状态通过若干操作到达的位置状态总数。

数据范围

- 对于 30% 的数据，$n, m \leq 5000$；
- 对于 100% 的数据，$n, m \leq 1000000$。

算法分析

对于第一个串的每个位置，不难直接求得第二个串上可能的最左位置和最右位置。但是，通过若干小数据可以发现，若两个序列的末尾两个字符分别为 XY 和 YX，则该状态显然无法到达。可以证明，不可到达的状态有且仅有上述情形，因此我们只需在扫描整个序列的同时减去所有特例即可。整个算法的时间复杂度为 $\Theta(n)$。

时间复杂度

$\Theta(n)$

空间复杂度

$\Theta(n)$

1.22 266D-BerDonalds (D8746,P5649)

题目大意

给定一张 n 个点 m 条边的带权无向连通图，要求选取图上任意一条边的任意实数位置作为中心，使得其到图中任意节点最短距离的最大值最小。

数据范围

- 对于 2% 的数据，$n \leq 5$；
- 对于 100% 的数据，$n \leq 200$，$c \leq 100000$，没有重边或自环。

算法分析

由于题中允许将中心放置在边上的任意实数位置，所以中心的选择存在以下两类情形：

- 中心恰好在图中某个节点上；
- 中心在某条边的内部。

对于中心就在节点上的情形，只需预处理出所有节点两两之间的最短路，然后暴力枚举即可。因此，我们主要考虑中心在边内部的情形。对于这一问题，我们考虑依次枚举每条边，计算当中心选取在当前边上时的最优答案。

- 假设当前枚举的边为 (u, v)，边 (u, v) 的长度为 len，中心 c 距离端点 u 的距离为 l（$0 \leq l \leq len$）。若记图中任意两点 x 和 y 之间的最短距离为 $d_{x,y}$，则对于图中任意节点 x，$d_{c,x} = min(d_{u,x} + l, d_{v,x} + len - l)$；
- 由于 $d_{u,x}+l$ 以及 $d_{v,x}+len-l$ 均为一次函数，则当 $l \leq \frac{len+d_{v,x}-d_{u,x}}{2}$ 时，$d_{c,x} = d_{u,x}+l$，否则 $d_{c,x} = d_{v,x}+len-l$。若将图中所有节点按照其所对应的分界点 $\frac{len+d_{v,x}-d_{u,x}}{2}$ 排序，则对于所有分界点不超过当前 l 值的节点均有 $d_{c,x} = d_{v,x} + len - l$，对于剩余所有节点均有 $d_{c,x} = d_{u,x} + l$；
- 若记 $L(l)$ 为当中心 c 距离端点 u 的距离为 l 时，所有分界点不超过 l 的节点 $d_{v,x} + len - l$ 的最大值，并记 $R(l)$ 为当中心 c 距离端点 u 的距离为 l 时，所有分界点超过 l 的节点 $d_{u,x} + l$ 的最大值，则易证，随着 l 的增大，$L(l)$ 单调不减，$R(l)$ 单调不增。同时，由于 $L(l)$ 和 $R(l)$ 覆盖了所有节点，因此当中心 c 距离端点 u 的距离为 l 时，答案即为 $max(L(l), R(l))$；
- 若选择两个相邻的分界点 l_1 和 l_2，则易知在区间 $[l_1, l_2]$ 上，$L(l)$ 和 $R(l)$ 均为一次函数，因此要使得两者中最大值最小，则 l 只可能取在区间的端点上或者两直线的交点处；
- 易知，$L(l)$ 和 $R(l)$ 只需排序后分别维护前缀和后缀最大值即可。

综合以上分析，枚举每条边后单次统计的代价为 $\Theta(n \log n)$，因此整个算法的时间复杂度为 $\Theta(mn \log n)$。

时间复杂度

$\Theta(mn \log n)$

空间复杂度

$\Theta(n + m)$

1.23 266E-More Queries to Array... (D8750,P5629)

题目大意

给定一个长度为 n 的整数序列 $\{a_i\}$，依次进行 m 次操作，操作类型包括：

- 将某一给定区间内的所有值全部修改为某一给定值；
- 给定一段区间 $[l,r]$，询问 $\sum_{i=l}^{r} a_i \times (i-l+1)^k$ 模 (10^9+7) 的值。

数据范围

- 对于 10% 的数据，$n \leq 1000$；
- 对于 30% 的数据，$n \leq 10000$；
- 对于 50% 的数据，$n \leq 50000$；
- 对于另外 10% 的数据，$n \leq 100000$，且询问操作中 $k \leq 2$；
- 对于 100% 的数据，$1 \leq n \leq 100000$，初始数组中 $0 \leq a_i \leq 10^9$，赋值操作中 $1 \leq l \leq r \leq n$，$0 \leq x \leq 10^9$，询问操作中 $1 \leq l \leq r \leq n$，$0 \leq k \leq 5$。

算法分析

算法一

容易发现，这是一个区间修改、区间查询的问题，因此考虑借助线段树来解决。对于线段树中的每个节点，若其所表示的区间为 $[l,r]$，则我们可以对于每一个 k 值维护 $\sum_{i=l}^{r} a_i \times (i-l+1)^k$。在合并两个相邻区间的信息时，只需将右侧区间值的表达式展开即可，单次操作代价为 $\Theta(k^2)$。在修改操作中，由于是对一整段区间进行赋值，因此需要预处理出所有前 i 个数的 j 次方和。整个算法时间复杂度为 $\Theta(k^2 m \log n)$，需要进行一定的常数优化才能通过本题。

算法二

在算法一中，可以对维护的值作出适当的变动。我们可以对每个区间 $[l,r]$，维护 $\sum_{i=l}^{r} a_i \times i^k$。通过这样的改变，可以将单次维护的代价降低为 $\Theta(k)$。在每次询问操作中，可以在查询结束之后再将结果展开计算即可。整个算法的时间复杂度为 $\Theta(km \log n)$。

时间复杂度

$\Theta(km \log n)$

空间复杂度

$\Theta(n)$

1.24 268D-Wall Bars (D8756,P5620)

题目大意

给定一根高度为 n 的竖直杆，要求在竖直杆上每个高度为正整数的位置处恰好向东南西北四个方向之一伸出一根水平杆。定义一个方案合法当且仅当满足以下条件：对于东南西北四个方向，要求至少存在一个方向上，使得该方向上最矮的横杆不高于 h，且最高的横杆不低于 $n-h+1$，同时必须保证该方向上任意相邻的两根横杆高度差不超过 h。求合法的方案总数。

数据范围

- $1 \leq n \leq 1000$，$1 \leq h \leq min(n, 30)$。

算法分析

对于本题，由于 h 的范围较小，因此考虑分别记录四个方向上的状态，然后借助 DP 来解决。首先，由于一旦某个方向上的两根横杆之间高度差超过 h，则该方向上无论之后如何放置都不再合法，因此一旦前一根该方向上的横杆与当前高度之差超过 h，则可以直接视作 0。其次，由于必然有一个方向上的前一根横杆就是当前高度，因此我们只需记录另外三个方向上的状态即可。

考虑设计 DP 的状态为 $dp[i][j][k][l][st]$，其中 i 表示当前高度，j、k 和 l 分别表示除了当前高度处横杆以外的另三个方向上前一根横杆与当前高度之差，st 则取值只有 0 或 1，表示当前高度处横杆所处的方向是否仍然合法。接下来考虑转移，分两种情形讨论：

- 情形一：若放置的下一根横杆与当前高度处横杆同向，则转移到的状态应为 $dp[i+1][j+1][k+1][l+1][st]$，因为合法性不会发生改变。其中，需要注意 j 为 0 的情形，此时有两种可能：一种情形是该方向上之前没有过横杆，则操作过后依然没有横杆；另一种情形是该方向已经不合法，则操作后依然不合法。因此，当 j 为 0 时应该保持不变，对于 k 和 l 也同理。另外，若 j、k 或 l 等于 $h-1$，则在转移到的状态中也应该相应地变成 0。
- 情形二：若放置的下一根横杆与当前高度处横杆不同向，则转移到的状态应为 $dp[i+1][st][k+1][l+1][j \mid\mid i < h]$，对于 k 和 l 也同理，因为若放置的下一根横杆合法，则只可能是该方向上的第一根横杆或者是到当前高度为止该方向一直保持合法。

借助上述的转移方式，则我们只需利用滚动数组即可解决空间开销的问题。整个算法的时间复杂度为 $\Theta(nh^3)$。

时间复杂度

$\Theta(nh^3)$

空间复杂度

$\Theta(h^3)$

1.25 269D-Maximum Waterfall (D8778,P5543)

题目大意

已知某人造瀑布由 n 块水平板和高度为 t 的墙壁组成。已知第 i 块水平板高度为 h_i，横坐标区间为 $[l_i, r_i]$。墙壁的顶端可以被看做从 $(-10^9, t)$ 到 $(10^9, t)$ 的水平板。同时，墙壁的底端可以被看做从 $(-10^9, 0)$ 到 $(10^9, 0)$ 的水平板。数据保证，任意两块水平板之间没有公共点。定义流水可以从第 i 块水平板流向第 j 块水平板当且仅当：

- $max(l_i, l_j) < min(r_i, r_j)$，即两块水平板的横坐标区间相交；
- $h_j < h_i$，即流水只能从高处往低处流动；
- 不存在 k 满足 $h_j < h_k < h_i$，且 (i, k) 与 (k, j) 均完全满足以上两个条件。

定义从 i 到 j 的水流量为 $min(r_i, r_j) - max(l_i, l_j)$，即两块水平板横坐标区间相交部分的长度。定义整个人造瀑布的水流量为单向水流路径上相邻的两块水平板之间水流量的最小值。求从墙壁顶端到底端的最大水流量。

数据范围

- 对 30% 的数据，$1 \leq n \leq 100$；
- 对 100% 的数据，$1 \leq n \leq 10^5$，$2 \leq t \leq 10^9$。

算法分析

对于本题而言，由于水平板互不相交，则显然我们可以按照高度值依次加入所有水平板。当加入某一特定的水平板时，我们需要扫描所有与其横坐标区间有交集，并且当前依然可见的水平板，则只需借助 set 即可进行维护。

通过上述过程，则显然我们可以构建出所有可转移关系。可以证明，这样的关系最多只有 $\Theta(n)$ 组，因为当前的所有可见水平板最多只会被完整地包含一次，并且每加入一块水平板，部分相交的情形最多只会增加两组。整个算法的时间复杂度为 $\Theta(n \log n)$。

时间复杂度

$\Theta(n \log n)$

空间复杂度

$\Theta(n)$

1.26 269E-String Theory (D8764,P5587)

题目大意

给定一个 $n \times m$ 的矩形，在左右边界上，每行的中点处都有一个点；在上下边界上，每列的中点处都有一个点。现在有 $n+m$ 条线段，每条线段都是以不同侧边界上的两个点所确定的，并且每个点都恰好使用了一次。题目要求通过若干次交换某两行（列）的操作，使得最终任意两条线段不相交。在每次交换的过程中，同时交换所选择的两行（列）上处在对应边界上的点，但不改变每个点与所连接的线段的对应关系。例如，下图就是一种合法的交换方式：

数据范围

- 对于 30% 的数据，$1 \leq n, m \leq 10^3$;
- 对于 100% 的数据，$1 \leq n, m \leq 10^5$;

算法分析

Step One

通过观察我们首先可以发现：在这样的一个矩形中，如果借助每条线段的两个端点所处边界的不同来分类，则所有线段可以被分为这样六类：

- Top-Left，简记为 TL 型。
- Top-Right，简记为 TR 型。
- Bottom-Left，简记为 BL 型。
- Bottom-Right，简记为 BR 型。
- Left-Right，简记为 LR 型。
- Top-Bottom，简记为 TB 型。

同时，由于所有的交换操作都不会改变任何一条线段的属性。因此，我们首先得到了该问题有解的一个必要条件，即：在最初的情形中，LR 型和 TB 型的线段不能同时出现。

Step Two

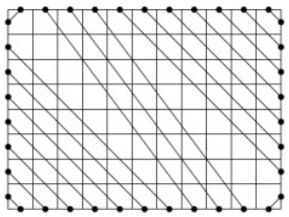
接下来，我们引入一些符号表示，以方便之后的叙述：

- 对于前面所提到的六类线段，我们分别使用 cnt_X 来表示 X 型的线段的数量；
- 对于边界上的每一个点，我们使用一个二元组 (x, y) 坐标来表示其所处的位置。其中，x 必定是 T、B、L、R 之一，以分别表示它处在上、下、左、右边界上；y 则是一个 $1 \sim n$（$1 \sim m$）的正整数，表示它所处的行（列）号；
- 对于每一条线段，我们使用其两个端点的坐标所构成的四元组来表示它的位置，我们暂且将这个四元组称之为该条线段的坐标。

下面，我们来证明：若该初始状态有解，则在最后的情形中，所有的点和线段的排布一定遵循这样的规则：

- 若存在 TL 型的线段，则它们依次排列在矩形的左上角，即所有 TL 型线段的坐标分别为 $(T, i, L, i), i = 1, 2, \ldots, cnt_{TL}$。
- 若存在 TR 型的线段，则它们依次排列在矩形的右上角，即所有 TR 型线段的坐标分别为 $(T, m - i + 1, R, i), i = 1, 2, \ldots, cnt_{TR}$。
- 若存在 BL 型的线段，则它们依次排列在矩形的左下角，即所有 BL 型线段的坐标分别为 $(B, i, L, n - i + 1), i = 1, 2, \ldots, cnt_{BL}$。
- 若存在 BR 型的线段，则它们依次排列在矩形的右下角，即所有 BR 型线段的坐标分别为 $(B, m - i + 1, R, n - i + 1), i = 1, 2, \ldots, cnt_{BR}$。
- 若存在 LR 型的线段，则它们依次排列在矩形纵向的中央，即所有 LR 型线段的坐标为 $(L, cnt_{TL} + i, R, cnt_{TR} + i), i = 1, 2, \ldots, cnt_{LR}$。
- 若存在 TB 型的线段，则它们依次排列在矩形横向的中央，即所有 TB 型线段的坐标为 $(T, cnt_{TL} + i, B, cnt_{BL} + i), i = 1, 2, \ldots, cnt_{TB}$。

例如，下图就是满足上述规则的一个例子：



通过上面这张图的辅助，我们不难验证：在矩形的每一侧边界上，同种类型的线段所连接的端点必然连续排列，否则不同种的线段之间必然存在交叉；同样，对于所有同类型的线段，只有相互平行的排列才能保证不出现交叉。因此，如果不考虑点、线段以及行和列的标号，则我们发现最终的方案其实是唯一的。

借助上面的结论，则我们又可以发现另一个初始方案有解的必要条件，即：$cnt_{TL} = cnt_{BR}$，并且 $cnt_{TR} = cnt_{BL}$。

Step Three

通过前两步的分析，我们就可以知道：如果初始方案有解，则最终所呈现出来的情形是怎样的。因此，现在我们只需要验证初始方案和最终的情形是否等价。要想完成这一步，则我们必须要借助在任何的交换操作中，始终保持不变的部分。

现在，我们把每个点当做是一张图的一个点，将每条线段当成是图中连接对应端点的一条实边。同时，我们将每个点与和它处在同一行（列）上的点之间用虚边相连接。例如，下图就是一张已经完成建边的图：

根据交换操作的性质，我们可以得出：在整个交换的过程中，这些点之间的所有边（包括实边和虚边）都不会发生任何改变。因此，我们验证两种情形是否等价的任务就变成了验证这两张对应关系图是否等价。同时，我们还可以发现这些构造出的对应关系图具有诸多特殊性：

- 这些点的属性实质上只与其所处的边界有关，而与其所处的具体行（列）号无关。因此，所有点都可以被归入 T、B、L、R 四类。
- 图中所有边构成了若干个互不相交的偶环，并且在所有的偶环中都是实边和虚边交错出现。

因此，要想判断两张对应关系图是否等价，也就转化成了判断这两张图中的所有环是否一一对应。

Step Four

由于在对应关系图中，所有的点仅有四类，所以单独判断两个环是否等价可以进行进一步的转化：把图中的每个点用代表它的属性的字符代替，则问题变为判断两个环形字符串是否等价，其中等价是指通过旋转以及翻转后完全相等。

对于这个问题，我们完全可以将两个字符串分别从任意一个位置断开，并将其中一个字符串翻倍，然后以另一个字符串作为模式串，对两个串执行字符串匹配算法。当然，对于翻转操作，则只需将翻倍的字符串整体倒置后，再次执行字符串匹配算法。

当问题转化到这一步后，我们还可以发现最后一个结论：如果一种情形中的一个字符串能够与另一种情形中的多个字符串相匹配，则无需考虑具体与哪一个对应，只需任选一个即可。这一点可以由每个点的属性只与其所处边界有关、而与具体行（列）号无关这一结论推出。

当我们得出上面的结论之后，我们便可以大胆地将两种情形中的所有环形字符串找出来，然后将其中一种情形的所有字符串作为模式串。最后，对另一种情形中的所有字符串依次与模式串进行匹配，若过程中出现某个串找不到模式串与其相匹配，则问题无解；否则，在找到相匹配的字符串后，我们只需要线性地扫描两个字符串所代表的对应关系图中的两个环，即可求出重新排列后各行（列）新的标号。

在具体实现中，可以直接暴力地找出初始状态和最终状态的所有环形字符串，然后对初始状态的所有字符串建立 $Aho-Corasick$ 自动机，并将最终状态的所有字符串依次进行匹配。整个算法的时间复杂度为 $\Theta(n+m)$。

时间复杂度

$\Theta(n+m)$

空间复杂度

$\Theta(n+m)$

1.27 273D-Dima and Figure (D9203,P5849)

题目大意

给定一张 $N \times M$ 的网格，要求选取出某些网格染成黑色，使得整张网格满足：

- 至少存在一个网格是黑色的；
- 所有的黑色网格形成一个四联通的连通块；
- 任意两个黑色网格之间的最短距离就是它们之间的曼哈顿距离。

求可能的染色方案总数。

数据范围

- 对于所有的数据，$1 \leq N, M \leq 150$。

算法分析

首先通过对限制条件的分析不难发现，所有的限制等价于要求整个黑色连通块构成一个凸多边形。因此，不妨设计 DP 状态为 $dp[i][j][k][st_1][st_2]$，表示当前处理到第 i 行，黑色网格区间的左右端点分别为 j 和 k，st_1 和 st_2 分别表示左右边界是单调不减的或者是单调不增的，则借助部分和之后转移显然。整个算法的时间复杂度为 $\Theta(NM^2)$。

时间复杂度

$\Theta(NM^2)$

空间复杂度

$\Theta(NM^2)$

1.28 273E-Dima and Game (D8796,P5504)

题目大意

初始时给定 n 个区间 (l_i, r_i)，两个玩家轮流进行操作，每次操作可以任意选择一个长度大于 2 的区间 (l_i, r_i)，并将其替换为区间 $(l_i + \lfloor \frac{r_i - l_i}{3} \rfloor, l_i + 2\lfloor \frac{r_i - l_i}{3} \rfloor)$ 或者是区间 $(l_i, r_i - \lfloor \frac{r_i - l_i}{3} \rfloor)$，不能进行操作的玩家为负。现要求构造合法方案，使得所有 l_i 和 r_i 均为不超过 p 的正整数，且先手必胜。求不完全相同的合法方案总数。

数据范围

- 对于 20% 的数据，$1 \leq n \leq 10$，$1 \leq p \leq 10$；
- 对于 100% 的数据，$1 \leq n \leq 1000$，$1 \leq p \leq 10^9$。

算法分析

首先，不难发现博弈过程中的状态只与每个区间的长度有关，而与具体的区间端点无关，因此显然可以对区间长度建立 SG 函数。通过打表可以发现该 SG 函数的增长是成段的，增长速度大概是 $\Theta(\log p)$ 级别的。因此，不妨通过递推成段地求出 SG 函数，然后便只需要一遍 DP 就能求出合法方案总数。整个算法的时间复杂度为 $\Theta(\log p + n)$。

时间复杂度

$\Theta(\log p + n)$

空间复杂度

$\Theta(\log p + n)$

1.29 274C-The Last Hole! (D8741,P5670)

题目大意

在一个无限大的二维平面中，给定 n 个点作为圆心，在任意时刻 t，所有圆的半径均为 t。同时，定义在该平面中，未被任何一个圆覆盖到的连通封闭区域为“洞”。求最后一个“洞”消失的时刻。

数据范围

- $1 \leq n \leq 100$，$-10^4 \leq x_i, y_i \leq 10^4$；
- 数据保证没有两个圆圆心相同。

算法分析

首先，易知在某个“洞”消失的时刻，其所处的点必然与至少三个圆心等距。其次，我们考虑证明，所有的“洞”只可能出现在某个锐角三角形的外心或者某个矩形的中心上。

- 考虑某个可能的“洞”的消失点 c，设与 c 等距的圆心分别为 $o_1, o_2, \ldots, o_m$（$m \geq 3$）；
- 若存在三个点 o_i、o_j 和 o_k，使得这三点构成一个锐角三角形，则 c 点是一种可能的答案，只需验证是否有其他点率先覆盖到点 c 即可；
- 若任意选出的三个点都只能构成钝角三角形，则 c 点必然不可能成为答案，因为其周围无法形成一个封闭的区域；
- 除去以上两种情形，则唯一的例外是剩余三个或四个点，且其中任意三点构成直角三角形。若只剩余三个点，则 c 点周围同样无法形成一个封闭的区域，而如果剩余四个点，则由条件可知，这四个点构成一个矩形；
- 因此，我们只需枚举所有锐角三角形的外心以及所有矩形的中心，然后依次进行验证即可。另外，由于矩形的特殊性，所以在枚举时可以列出所有点对，按照中点坐标以及两点间距离进行判断。

基于以上分析，我们只需暴力枚举所有可能的解即可解决本题，整个算法的时间复杂度为 $\Theta(n^4)$。

时间复杂度

$\Theta(n^4)$

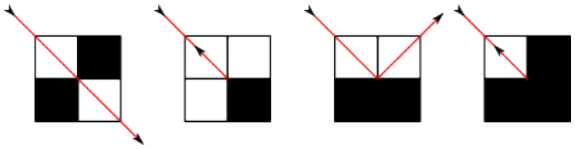
空间复杂度

$\Theta(n^2)$

1.30 274E-Mirror Room (D9209,P5768)

题目大意

给定一张 $n \times m$ 的网格，其中共有 k 个障碍格，剩余网格均为空。已知一束激光的起点为 (x_s, y_s)，初始方向为东北、西北、东南、西南之一。已知激光碰到障碍或者边界时的反射情况如下：



求至少被通过一次的网格总数。

数据范围

- 30% 的数据，$n, m \leq 10$；
- 60% 的数据，$n, m \leq 1000$；
- 另 10% 的数据，$k = 0$；
- 100% 的数据，$n, m, k \leq 100000$。

算法分析

首先不难发现，由于激光在 $\Theta(n+m+k)$ 次反射内必然陷入循环，因此我们只需统计其中一个完整的周期即可。对于这一问题，我们不妨将主对角线方向和副对角线方向上的路径分开统计，则只需在每条对角线上维护区间并即可。同时，我们还需考虑不同方向对角线的重复计数问题，但是可以证明，在题中所给的反射规则之下，不可能出现重复计数问题，因为若对整张网格进行黑白染色，则在主对角线和副对角线之间的每次转换都会改变所在网格的颜色，因此同一个网格不可能同时被两种方向的对角线通过，因此我们只需要将两个方向上的贡献累加即可。整个算法的时间复杂度为 $\Theta((n+m+k)\log(n+m))$。

时间复杂度

$\Theta((n+m+k)\log(n+m))$

空间复杂度

$\Theta(n+m+k)$

1.31 277D-Google Code Jam (D8804,P5492)

题目大意

已知某场考试总时间为 t，共有 n 道题，每道题的小数据价值 $scoreSmall_i$ 分，解题需要 $timeSmall_i$ 的时间，大数据价值 $scoreLarge_i$ 分，解题需要在做完小数据的前提下多花 $timeLarge_i$ 的时间，但有 $probFail_i$ 的概率会 FST。求可能的最大期望得分，以及在最大期望得分的前提下的最小期望罚时。其中，定义罚时为最后一次成功提交的时间。

数据范围

- $1 \leq n \leq 1000$，$1 \leq t \leq 1560$；
- $1 \leq scoreSmall_i, scoreLarge_i \leq 10^9$，$1 \leq timeSmall_i, timeLarge_i \leq 1560$，$0 \leq probFail_i \leq 1$。

算法分析

首先，为了便于进行 DP，不妨将所有问题小数据部分的 $probFail_i$ 值全部看做为 0。其次，若已经确定了所有待解决的问题，并且已知解决这些问题的顺序，则期望的总得分为 $\sum_i score_i(1 - probFail_i)$。同时，期望的罚时则显然可以通过递推来解决。

接下来，若已经确定了所有待解决的问题，但并未确定解决这些问题的顺序，则通过归纳法可以证明，解决这些问题的顺序必然是按照 $\frac{probFail_i}{1-probFail_i} time_i$ 从小到大的顺序。另外，不难发现通过上述的排序方式，则每道题的小数据部分显然排在大数据部分之前。

最后，在题设的情形中，则我们只需要首先按照上述方式对问题进行排序，然后再执行一遍类似于背包的 DP 即可解决，转移方程较为显然。整个算法的时间复杂度为 $\Theta(nt)$。

时间复杂度

$\Theta(nt)$

空间复杂度

$\Theta(nt)$

1.32 280D-k-Maximum Subsequence Sum (D8752,P5627)

题目大意

给定一个长度为 n 的整数序列，进行 m 次操作，操作的类型包括：

- 修改某个给定位置上整数的值；
- 询问在某一段给定区间中，取出不超过 k 段的最大不相交子段和。

数据范围

- $1 \leq n, m \leq 10^5$，序列元素绝对值不超过 500，询问操作中的 k 不超过 20。

算法分析

算法一

显然，对于这样一个单点修改、区间查询的问题，考虑使用线段树直接维护。对于线段树中的每个节点，若记其维护的区间为 $[l, r]$，则我们可以直接维护下列一些值：

- 在 $[l, r]$ 中，取恰好 i 段的最大不相交子段和；
- 在 $[l, r]$ 中，必须选取区间的左端点 l，取恰好 i 段的最大不相交子段和；
- 在 $[l, r]$ 中，必须选取区间的右端点 r，取恰好 i 段的最大不相交子段和；
- 在 $[l, r]$ 中，必须同时选取区间的左右端点 l, r，取恰好 i 段的最大不相交子段和。

通过维护的这些值，每个节点单次维护的代价均为 $\Theta(k^2)$。因此，整个算法的时间复杂度为 $\Theta(k^2 m \log n)$。在极限情形下，若不经过大量常数优化，该算法仍难以通过本题。

算法二

在该题中，我们可以换一种完全不同的思路。考虑将 1 到 $n+1$ 作为节点，并新建源点 s 和汇点 t。在该网络流模型中，考虑按如下方式建边：

- $\forall i$，建边 $(i, i+1)$，容量为 1，费用为序列中的第 i 个整数值；
- $\forall i$，建边 (s, i) 以及 (i, t)，容量均为 1，费用均为 0。

对于每次询问，若再对源点 s 附加限制其流出的流量最大为 k，则该图中的最大费用最大流即为该次询问的答案。然而，由于单次网络流算法的代价过高，所以我们需要充分利用其中的特性。

算法三

在算法二中，不难发现在每次询问中，每一单位的流量都流经了一段连续的区间，其所产生的费用为该段区间所有整数值之和，并且在此之后，该段区间中的费用全部被取负。因此，我们可以继续使用线段树来模拟整个过程。对于线段树中的每个节点，只需维护区间的最大子段和。另外，由于存在取负标记，所以还需维护区间的最小子段和。在具体实现时，由于每次询问操作之后要进行复原，所以对于最大（小）子段和，必须同时记录下其对应的方案。整个算法的时间复杂度为 $\Theta(km \log n)$。

时间复杂度

$\Theta(km \log n)$

空间复杂度

$\Theta(n)$

1.33 283E-Cow Tennis Tournament (D8767,P5582)

题目大意

给定 n 头能力值互不相同的奶牛，初始时任意两头奶牛之间的比赛结果均为能力值高的赢。给定 k 次操作，每次操作 a_i b_i（$a_i < b_i$）表示，将所有参赛双方奶牛的能力值均在区间 $[a_i, b_i]$ 内的比赛结果全部改为相反。求所有操作结束后，在所有奶牛中存在多少个无序三元组 (x, y, z)，使得 x 赢 y、y 赢 z、z 赢 x。

数据范围

- $3 \leq n \leq 10^5$，$0 \leq k \leq 10^5$；
- $1 \leq s_i \leq 10^9$，$1 \leq a_i < b_i \leq 10^9$。

算法分析

考虑这 n 头奶牛所能构成的所有三元组不难发现，在所有操作结束后，除去题目中所要求统计的环形胜负关系，则剩余的所有三元组 (x, y, z) 中，必然有且仅有一头奶牛能战胜另两头奶牛，并且有且仅有一头奶牛会同时负于另两头奶牛，剩下的另一头奶牛则必然是一胜一负。基于这一特征，我们只需统计所有非环形胜负关系的三元组便同样可以得出答案。另外，我们还可以进一步发现，若对于某一头奶牛 x，任意选出两头均会被 x 打败的奶牛 y 和 z，则三元组 (x, y, z) 必定是我们所需统计的，且所有情形均不会被遗漏或者重复计数。因此，我们可以将题目转化为依次统计每头奶牛所能战胜的奶牛的数量。

对于这一问题，我们首先将所有奶牛按能力值从小到大排序，然后尝试对某一头特定的奶牛 x 进行研究：

- 记 $win_x[i]$ 为奶牛 x 对奶牛 i 的胜负情况，其中 1 表示 x 能战胜 i，0 则表示 x 负于 i。那么在所有操作进行之前，对所有 $i < x$ 有 $win_x[i] = 1$，对所有 $i > x$ 有 $win_x[i] = 0$；
- 根据题目定义，所有可能对奶牛 x 的比赛结果造成影响的操作必然满足 $a_i \leq x \leq b_i$，且其具体表现为将区间 $[a_i, b_i]$ 的值全部取反。在所有操作结束后，奶牛 x 所能战胜的奶牛总数便是 $win_x[i]$ 的总和；
- 对于这一区间取反、区间求和的问题，不难想到借助线段树来解决。由于取反操作的可逆性，因此当统计的对象从奶牛 x 变为奶牛 $x + 1$ 时，无论是左侧剔除的操作区间还是右侧新加入的操作区间，均可以视为对区间 $[a_i, b_i]$ 的取反。另外，由于初始状态的差异，每次查询前还需对当前统计对象 x 单独取反一次。

通过上述分析，在离散化之后，我们便只需在线段树上维护取反标记即可解决本题，整个算法的时间复杂度为 $\Theta((n + k) \log n)$。

时间复杂度

$\Theta((n + k) \log n)$

空间复杂度

$\Theta(n)$

1.34 285E-Positions in Permutations (D9173,P5996)

题目大意

对于一个 $1 \sim n$ 的给定排列 P，定义位置 i 是完美的当且仅当 $|P_i - i| = 1$。求长度为 n 且完美位置的总数恰好为 k 的方案数。

数据范围

- $1 \leq n \leq 1000$，$0 \leq k \leq n$。

算法分析

对于这一问题，显然考虑使用 DP 来解决，不妨设计 DP 状态为 $dp[i][j]$，表示前 i 个位置中有恰好 j 个完美位置的方案总数，则对于当前位置 i 而言，除去选择 $i-1$ 或者 $i+1$ 两种情形，剩余所有情形均是等价的，因为我们可以在所有完美位置全部放置完成后，对于剩余所有数直接随机放置。然而，由于转移到当前位置 i 时 $i-1$ 可能已经被使用，因此我们需要将状态修改为 $dp[i][j][st1][st2]$，其中 $st1$ 和 $st2$ 均为布尔值，分别表示 i 和 $i+1$ 是否被使用过。

再次观察上述 DP 过程，可以发现由于剩余的所有数是随机排列，因此会出现重复计数的问题，所以其实状态 $dp[i][j][st1][st2]$ 表示的是前 i 个位置中至少有 j 个完美位置的方案总数。易证，完美位置数为 k 对 j 的贡献为 C_j^k，则只需预处理组合数后进行容斥即可。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n^2)$

1.35 286D-Tourists (D8757,P5619)

题目大意

在给定的平面直角坐标系中，已知有 m 堵墙，每堵墙会从时刻 t_i 开始出现在 $[l_i, r_i]$。给出 n 组询问，每组询问给定出发时刻 q_i，求在时刻 q_i 从原点出发，以单位速度移动，则有多长时间会见到墙。

数据范围

- $1 \leq n, m \leq 10^5$，$0 \leq l_i < r_i \leq 10^9$，$0 \leq t_i \leq 10^9$，$0 \leq q_i \leq 10^9$。

算法分析

首先，对于每个询问显然可以单独处理，因此我们只考虑出发时刻为 t 时的情形。易知，在坐标为 x 处遇到墙的充要条件是 $t + x \geq c_x$，其中 c_x 表示坐标为 x 处墙最早的出现时刻。因此，遇到墙的总时间就等价于满足 $t \geq c_x - x$ 的坐标个数。

对于这一问题，显然我们可以通过预处理得到所有极大的区间 $[L_i, R_i]$，使得 $[L_i, R_i]$ 内的所有坐标处墙最早的出现时间均为 T_i。然后，我们可以借此统计出对于所有坐标处 $c_x - x$ 值的总出现次数。具体而言，一段区间 $[L, R]$ 的贡献为将 $[T - R, T - L]$ 内所有值的个数均加一，而单次的询问转化为查询小于等于 t_i 的数的总个数，所以只需使用线段树即可解决。整个算法的时间复杂度为 $\Theta((n + m) \log m)$。

时间复杂度

$\Theta((n + m) \log m)$

空间复杂度

$\Theta(m \log m)$

1.36 286E-Ladies' Shop (D9220,P5734)

题目大意

给定 n 个承重各不相同的背包，以及一个限重值 m。要求设计若干种重量各不相同的物品，使得这些物品满足：假设每种物品有无数件，则对于任意一个背包，都必须存在一种选取物品的方案，使得其总重量恰好为背包的承重；同时对于任意一种选取物品的方案，若其总重量不超过限重值 m，则必须存在一个背包，使得该背包恰好能装下该种方案的所有物品。另外，题目要求在满足要求的前提下，求出一组物品种数最少的方案。

数据范围

- 对于 40% 的数据，$n \leq 100$。
- 对于 100% 的数据，$1 \leq n, m \leq 10^6$。

算法分析

首先，由于要求所有总重量不超过 m 的选取物品的方案都必须有对应的背包，则所有单个物品都必须有背包与其对应。因此，在最后构造出的方案中，所有物品的重量必定是所提供的所有背包的承重的一个子集。

接下来，我们可以先解决是否存在合法解的问题。显然，由于每种物品都有无数个，所以要使得问题有解，则即使某个背包的承重所对应的物品未被选取，也必定存在一些物品的组合使得其总重量恰为该背包的承重，这就相当于存在了对应重量的物品。因此，要使得问题有解，则所有选取物品方案的总重量就是所有背包的承重可能组合出的重量。于是，我们只需要判断所有背包的承重可能组合出的重量是否全部合法。对于这一问题，我们又可以进一步发现：我们只需要判断任意两个重量的和是否合法即可。对于这一点，其必要性是显然的，而其充分性有如下的归纳证明：假设当前所有进行了不超过 k 次组合的方案都有背包对应，这表明经过不超过 k 次组合，我们所能得到的不超过 m 的所有重量还是所有初始的重量，那么当我们进行再一次的组合时，所得到的所有重量依然不会发生改变，即所有进行了不超过 $k+1$ 次组合的方案都有背包对应。

通过上面的分析，我们只需判断任意两个重量的和是否有背包对应。同时，在这过程中，若某个重量可以由另两个重量相加得到，则该重量的物品可以不必选取，否则该重量的物品必须选取。因此，问题变成了判断这些重量两两相加可以得到哪些重量。对于这一问题，我们可以构造一个 m 次的多项式，若存在重量为 i 的背包，则该多项式的第 i 项系数为 1，否则系数为 0。然后运用 $Fast\ Fourier\ Transformation$ 得到该多项式自乘以后的结果。这样，我们便可以通过最后的多项式中第 i 项的系数是否为 0 来判断重量 i 是否可以由两个已有重量相加得到。整个算法的时间复杂度为 $\Theta(m \log m)$。

时间复杂度

$\Theta(m \log m)$

空间复杂度

$\Theta(m)$

1.37 293B-Distinct Paths (D9189,P5938)

题目大意

给定一张 $n \times m$ 的网格，其中某些网格已经被染成 k 种颜色中的某一种。要求用这 k 种颜色将所有网格染色，使得从左上角到右下角的所有路径上均不存在同色的网格。同时，已知一条路径只能向右或向下走，求可能的染色方案总数。

数据范围

- 对于 50% 的数据，$1 \leq n, m \leq 3$；
- 对于 100% 的数据，$1 \leq n, m \leq 1000$，$1 \leq k \leq 10$。

算法分析

首先，易知从左上角到右下角的每条路径上均经过了恰好 $n+m-1$ 个网格，所以至少要有 $n+m-1$ 种不同的颜色。因此，根据 k 的大小可以极大地限定网格的大小。

由于可能的网格范围很小，因此考虑直接进行搜索。通过简单的分析可以发现，其实颜色的具体标号是没有意义的，我们只需搜索颜色的相对关系即可，同类型的方案数可以直接计算得到。因此，我们可以在搜索时按照颜色出现的先后顺序进行重标号，对于已经存在颜色的网格则枚举其颜色对应的映射。可以发现这样搜索得到的状态总数非常少，所以整个算法可以很快得出答案。

时间复杂度

$\Theta(k^{nm})$

空间复杂度

$\Theta(nm)$

1.38 293D-Ksusha and Square (D9162,P6047)

题目大意

给定一个面积不为 0 的凸多边形，要求在凸多边形的内部或边界上随机选取两个整点，求以这两个整点间连线为一条对角线的正方形面积的期望。

数据范围

- 对于 30% 的数据，$3 \leq n \leq 30$，坐标的绝对值 ≤ 100；
- 对于 100% 的数据，$3 \leq n \leq 10^5$，坐标的绝对值 $\leq 10^6$；
- 请注意，题目并没有保证没有三点共线。

算法分析

不难发现，正方形的面积等于两点间距离平方的一半，因此显然横纵坐标的贡献相互独立。接下来仅考虑横坐标的贡献，我们需要求出所有凸包内整点两两之间横坐标差的平方和。其次，易知同一横坐标处所有点的贡献是完全等价的，因此我们只需要统计每个横坐标处凸包内整点的个数。对于这一问题，由于凸包上点的坐标范围较小，因此我们可以直接根据凸包上的边界暴力求解即可。整个算法的时间复杂度为 $\Theta(n+l)$，其中 l 表示凸包上点的坐标范围。

时间复杂度

$\Theta(n+l)$

空间复杂度

$\Theta(n+l)$

1.39 293E-Close Vertices (D8783,P5535)

题目大意

给定一棵 n 个节点的带权无根树，求树上所有距离不超过 L，且边权和不超过 W 的简单路径总数。

数据范围

- 共包含 20 个测试数据，每个测试点 5 分；
- *case* $1 \ldots 6$，满足 $n \leq 1000$；
- *case* $7 \ldots 10$，满足所有边的边权为 0；
- *case* $11 \ldots 16$，满足树构成一条链；
- 所有数据满足，$1 \leq n \leq 100000$，$1 \leq L \leq n$，$1 \leq W \leq 10^9$，所有边权小于 10^4，且 $p_i \leq i$。

算法分析

本题显然是一道点分治的模板题。在每次分治的过程中，我们需要求出所有经过重心的合法路径。对于这一问题，由于合法路径需要满足两种权值的限制，则我们显然可以先按照到重心的边权和排序，则在处理过程中，有效区间的端点必然是单调的。同时，若不考虑同一棵子树的问题，则只需借助树状数组来统计即可。若考虑重复计数的问题，则其实只需对每棵子树分别再执行一次类似的过程即可。整个算法的时间复杂度为 $\Theta(n \log^2 n)$。

时间复杂度

$\Theta(n \log^2 n)$

空间复杂度

$\Theta(n)$

1.40 301C-Yaroslav and Algorithm (D8810,P5473)

题目大意

给定一个算法，描述如下：

- 输入一个正整数 a，将其看做字符串；
- 给定一些命令形如 $s_i>>w_i$ 或者 $s_i<>w_i$，其中 s_i 与 w_i 均为长度不超过 7 的可以为空的字符串，只能由数字与字符 ? 组成；
- 每次操作寻找一个编号最小的命令 i，使得 s_i 是字符串 a 的子串，若不存在则结束算法；
- 对于命令 i，找到 s_i 在字符串 a 中第一次出现的位置，并将其替换为 w_i；若形式为 $s_i>>w_i$，则算法继续执行，若形式为 $s_i<>w_i$，则算法结束；
- 算法结束时 a 的值就是算法的输出。

现在给定 n 个正整数，要求设计一组命令集合，使得每个数在执行一次上述算法后值在十进制下加一。要求命令条数不超过 50 条，且每次算法的执行步数不超过 200 步。

数据范围

- $1 \leq$ 每个数 $\leq 10^{25}$。
- 共有 20 个测试点，对于第 i 个测试点，$n = 5i$。

算法分析

由于本题要求对输入的 n 个数算法均能正确执行，因此我们考虑借助 ? 设计一种具有通用性的算法。对于这一问题，我们可以模拟进位加法的整个过程，并用 ? 来表示当前处理到的位置。另外，由于算法中的每次操作均是从左往右寻找匹配位置，因此还需利用 ? 和 ?? 来判断方向。下面的方案就是一组可行的解：

0?? <> 1
1?? <> 2
2?? <> 3
3?? <> 4
4?? <> 5
5?? <> 6
6?? <> 7
7?? <> 8
8?? <> 9
9?? >>??0
??0 <> 10
?0 >> 0?
?1 >> 1?
?2 >> 2?
?3 >> 3?
?4 >> 4?
?5 >> 5?
?6 >> 6?
?7 >> 7?
?8 >> 8?
?9 >> 9?
0? <> 1
1? <> 2
2? <> 3
3? <> 4
4? <> 5
5? <> 6
6? <> 7
7? <> 8
8? <> 9
9? >>??0
1 >>?1
2 >>?2
3 >>?3
4 >>?4
5 >>?5
6 >>?6
7 >>?7
8 >>?8
9 >>?9

整个算法的时间复杂度为 $\Theta(1)$。

时间复杂度

$\Theta(1)$

空间复杂度

$\Theta(1)$

1.41 301E-Yaroslav and Arrangements (D8739,P5689)

题目大意

若某个数列的任意相邻两项（包括首尾两项）之差均为 1，且第一项为此数列中的最小值，则认为该数列合法。要求构造一个单调不下降的数列 $\{b_i\}$，使得其长度不超过 n，数列中的数均为 m 以内的正整数，同时要求重新排列该数列，可以得到至少 1 个至多 k 个合法的序列。求数列 $\{b_i\}$ 可能的方案数。

数据范围

- 30% 的数据，$n, m, k \leq 8$；
- 60% 的数据，$n, m, k \leq 60$；
- 100% 的数据，$n, m, k \leq 100$。

算法分析

本题较易想到通过 DP 的方式进行计数，因此我们考虑设计状态。首先易知当前 $\{b_i\}$ 数列的长度 i 以及数列的最后一个数字 j 必须要记录。其次通过观察，一个合法的序列必然可以通过如下方式得到：

- 首先确定序列中的最小值，即合法序列中的第一个值，并将该值复制若干份；
- 接下来，假设当前已经进行到数 j，则考虑向该序列中插入若干个数 $j+1$。易知，所有连续的数 $j+1$ 必然是插在两个之前相邻的数 j 之间，且所有相邻的数 j 之间都必须插入至少 1 个数 $j+1$。

因此，我们还需记录当前至少需要多少个数 $j+1$ 来填补空当，不妨记为 p。另外，由于题中还有对于可能的方案数的限制，所以还要追加记录形成当前状态的方案数为 q。

通过上述记录的这些值，我们便已经可以开始进行转移。若记 $dp[i][j][p][q]$ 为满足当前状态的数列 $\{b_i\}$ 的方案数，假设增加 l（$l \geq p$）个数 $j+1$ 到排列中，转移到的状态为 $dp[i'][j'][p'][q']$，则不难证明：

- $i' = i + l$，$j' = j + 1$，$p' = l - p$，$q' = q \times F(l-p, p)$；
- 其中 $F(a, b)$ 表示不定方程 $\sum_{i=1}^{b} \mathcal{X}_i = a$ 的非负整数解的组数，易知 $F(a, b) = C(a + b - 1, a)$；
- 若当前状态中的 $p = 0$，则表示该方案已经结束构造，并且应该被计入答案当中。

另外，在具体实现时，可以有如下一些优化：

- 限制当前已有的数字和需求的下限之和不超过长度限制 n；
- 可以将所有方案中的最小值均设为 1，因为其余类似的方案均可在统计答案时一起计数；
- 易知当序列长度为 n 且最小值为 1 时，所用的数字最多不会超过 $\lfloor \frac{n}{2} \rfloor + 1$。

结合上述的一些优化即可解决本题，整个算法的时间复杂度为 $\Theta(n^3 mk)$。

时间复杂度

$\Theta(n^3mk)$

空间复杂度

$\Theta(n^2mk)$

1.42 303D-Rotatable Number (D8808,P5485)

题目大意

要求找到最大的 $b(1 < \mathrm{b} < \mathrm{x})$，满足在 b 进制下存在长度为 n 的正旋转数（允许有前导零）。

数据范围

- 对于 20% 的数据，$n \leq 10$，$x \leq 15$；
- 对于 50% 的数据，$x \leq 10$；
- 对于 100% 的数据，$1 \leq n \leq 5 \times 10^6$，$2 \leq x \leq 10^9$。

算法分析

本题是一个结论题，具体解释可以参见维基百科。根据该结论可知，b 进制下有解当且仅当 $n+1$ 为质数，且 b 是模 $n+1$ 意义下的一个原根。因此，问题转化为寻找模 $n+1$ 意义下 x 以内最大的原根，对于这一问题则只需暴力枚举然后进行验证即可。整个算法的时间复杂度为 $\Theta(d(n+1)\log(n+1))$。

时间复杂度

$\Theta(d(n+1)\log(n+1))$

空间复杂度

$\Theta(d(n+1))$

1.43 303E-Random Ranking (D9186,P5943)

题目大意

已知某场考试共有 n 名考生，给定第 i 名考生可能的得分区间为 $[L_i, R_i]$（得分可以是任意实数值）。求每名考生得到每种排名的概率。

提示：无需考虑多名考生具有相同分数的情形。

数据范围

- 对于 10% 的数据，$1 \leq n \leq 2$；
- 对于 30% 的数据，$1 \leq n \leq 20$；
- 对于 50% 的数据，$1 \leq n \leq 50$；
- 对于 100% 的数据，$1 \leq n \leq 80$，$0 \leq L_i < R_i \leq 10^9$；
- 保证数据有梯度；
- 与标准输出相差 10^{-6} 以内算做正确。

算法分析

对于本题而言，首先不难发现，若第 i 名考生的得分位于区间 $[a, b]$ 内，且同时还有另外 k 名考生的得分都在区间 $[a, b]$ 内，则第 i 名考生在这些考生中得到任意一种排名的可能性完全相等。

借助上述结论，我们可以首先离散化所有的得分区间，然后考虑对于第 p 名考生，分别枚举最终所处的得分区间，则设计 DP 状态为 $dp[i][j]$，表示在当前有 i 名考生得分低于枚举的得分区间，且有 j 名考生得分处于枚举的得分区间中的情形下，出现该种状态的概率，则转移显然。

在一轮 DP 结束之后，则 $dp[i][j]$ 对于考生 p 排名为 $i+1, i+2, \ldots, i+j+1$ 都有 $\frac{dp[i][j]}{j+1}$ 的贡献。整个算法的时间复杂度为 $\Theta(n^5)$。

时间复杂度

$\Theta(n^5)$

空间复杂度

$\Theta(n^2)$

1.44 305D-Olya and Graph (D9214,P5760)

题目大意

给定一张 n 个点 m 条边的有向无权图，要求添加若干条边，使得新图满足：

- 对于任意一条有向边 (u,v)，必有 $u<v$；
- 以点 i 为起点，可以到达 $i+1,i+2,\ldots,n$ 的所有点；
- 任意两点间最多只有一条有向边；
- 对于任意点对 (i,j)（$i<j$），若 $j-i\leq k$，则从 i 到 j 的最短距离为 $j-i$；
- 对于任意点对 (i,j)（$i<j$），若 $j-i>k$，则从 i 到 j 的最短距离为 $j-i$ 或者 $j-i-k$。

求满足上述所有条件的加边方案数。

数据范围

- 保证输入中任意一对点 u_i,v_i 之间最多有一条边。并且保证输入中给出的 u_i 不下降。如果有多条有向边从 u_i 出发，那么这些边将按照 v_i 升序给出。
- 对于 30% 的数据，保证 $n,m\leq 4$；
- 对于 60% 的数据，保证 $n,m\leq 10^4$；
- 对于 100% 的数据，$2\leq n\leq 10^6$，$0\leq m\leq 10^5$，$1\leq k\leq 10^6$。

算法分析

首先由于前两条性质，则所有有向边 $(i,i+1)$ 必须存在。其次，结合最后两条性质可知，剩余的所有有向边都只能形如 $(i,i+k+1)$。同时，若存在 i 和 j，使得有向边 $(i,i+k+1)$ 和 $(j,j+k+1)$ 均存在，且满足 $i+k+1\leq j$，则方案必然不合法。因此，所有可行方案中跨度为 $k+1$ 的有向边的起点必然只会出现在某个长度为 k 的区间中。结合以上分析，则只需枚举第一条跨度为 $k+1$ 的有向边的起点即可，区间内的其余起点随意是否选取。整个算法的时间复杂度为 $\Theta(n)$。

时间复杂度

$\Theta(n)$

空间复杂度

$\Theta(n)$

1.45 305E-Playing with String (D8802,P5494)

题目大意

初始时给定一个字符串 s，两个人轮流操作，每次操作可以任意选择一个字符串，然后把该字符串的某个字符去掉，并将该字符左右两侧剩下的部分分别作为一个新的字符串。同时，对于每次操作所选取的字符，规则中要求必须是其所在字符串中，某个长度为奇数且大于 1 的回文串的回文中心。询问在双方都采取最优策略的情况下，先手是否必胜，若必胜则求出第一步操作可能的最小下标。

数据范围

- 对于 25% 的数据，$1 \leq |s| \leq 10$。
- 对于 50% 的数据，$1 \leq |s| \leq 100$。
- 对于 75% 的数据，$1 \leq |s| \leq 1000$。
- 对于 100% 的数据，$1 \leq |s| \leq 5000$。

算法分析

显然，一个字符是某个长度为奇数且大于 1 的回文串的回文中心的充要条件，是该字符是一个长度为 3 的回文串的回文中心，即该字符左右两侧的字符相同。因此一个字符是否可以成为决策点和其所处的字符串无关，只由其左右两侧的字符决定。借助这一点，我们可以预处理出所有可能的决策点。同时，若某次选择了一个决策点，则其左右两侧相邻的位置都不再可能成为决策点。

最后，我们发现决策点组成了一段段连续的区间，且任意两段决策点区间互不影响。同时，由于每段决策点区间的结果只与其长度有关，所以我们可以直接套用 $Sprague - Grundy$ 函数来解决本题。

在预处理时，我们计算出所有长度的区间所对应的 $Sprague - Grundy$ 值，然后取所有决策点区间长度所对应 $Sprague - Grundy$ 值的异或值，以此来判断是否先手必胜。对于题中的第二问，我们只需暴力枚举第一步的决策点，通过重新计算去掉该处字符后的 $Sprague - Grundy$ 值来判断该处是否可以成为第一步的决策点即可。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n)$

1.46 306C-White,Black and White Again (D8784,P5526)

题目大意

已知有 w 件好事和 b 件坏事将在接下来的 n 天内依次发生。要求每天至少发生一件事，且每天发生的事件要么全是好事要么全是坏事。同时，要求所有坏事只能发生于连续的若干天内。另外，事件发生的顺序不同也认为是不同的方案。求不同的方案总数。

数据范围

- $3 \leq n \leq 4000$；
- $2 \leq w \leq 4000$；
- $1 \leq b \leq 4000$；
- $w + b \geq n$。

算法分析

对于本题，由于 n 的范围非常小，因此可以直接枚举 i 和 j，表示坏事发生在第 i 天到第 j 天，则其对应的方案数为 $C_{b-1}^{j-i}C_{w-1}^{n-j+i-2}$。另外，由于要考虑事件发生的顺序，因此还需再乘上 $w!b!$。综合以上分析，显然我们只需预处理出所有阶乘对应的逆元即可 $\Theta(1)$ 回答。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n)$

1.47 306D-Polygon (D9235,P5503)

题目大意

给定点数 n，要求构造一个 n 个顶点的凸多边形，使得凸多边形的所有内角全部相等，并且每条边的长度均不相同。要求凸多边形的边长均为区间 $[1, 1000]$ 内的实数，顶点坐标的绝对值不超过 10^6。求任意一组合法方案。

数据范围

- $3 \leq n \leq 100$。

算法分析

- 通过简单的数学知识可知，当 $n = 3$ 或者 $n = 4$ 时必然不存在合法解。对于 $n > 4$ 的情形，对于第 i（$1 \leq i < n$）条边，不妨直接令其边长为 $500 - 0.01i$，则显然这些边长两两互不相同。而对于最后一条边，通过实践可以发现，利用上述方式构造出的凸多边形的最后一条边也必然合法。因此，在整个构造过程中，我们只需要借助向量运算来旋转一定的角度即可。整个算法的时间复杂度为 $\Theta(n)$。

时间复杂度

$\Theta(n)$

空间复杂度

$\Theta(1)$

1.48 309B-Context Advertising (D9201,P5884)

题目大意

给定一篇由 n 个单词组成的文章，每个单词的长度均已知。要求从中按顺序截取出若干个连续的单词，放置在 $r \times c$ 的广告板上，且必须满足：

- 单词的相对顺序不能改变；
- 同一单词不能跨行分布；
- 同一行内相邻的单词之间必须用一个空格隔开。

求在满足上述所有条件的前提下，最多能截取出多少个单词。

数据范围

- 30% 的数据保证：$n \leq 1000$；
- 100% 的数据保证：$1 \leq n, r, c \leq 10^6$，$r \times c \leq 10^6$。

算法分析

显然，根据单调性可知，我们可以先预处理出以每个单词为某一行的第一个单词，则最多能放置到哪个单词。其次，每一行均贪心地放置尽可能多的单词的贪心策略必然正确。因此，题中所求即为以某个位置为起点并向后延伸至多 r 次所能截取的最大长度。

对于这一问题，不妨将其看做一种树结构，则所求即为某一点的 r 层祖先，因此只需通过倍增的方式即可解决。整个算法的时间复杂度为 $\Theta(n \log n)$。

时间复杂度

$\Theta(n \log n)$

空间复杂度

$\Theta(n \log n)$

1.49 309D-Tennis Rackets (D8769,P5558)

题目大意

给定一块正三角形网球拍，已知每条边上的所有 n 等分点，且要求所有满足距离某一个顶点不超过 m 的等分点均不可以使用。求在三条边上各选出一个合法点并且构成的三角形为钝角三角形的方案数。

数据范围

- 对于 30% 的数据：$n \leq 1001$；
- 对于 100% 的数据：$1 \leq n \leq 32000$，$0 \leq m \leq \frac{n}{2}$。

算法分析

不难发现，若已知了该钝角三角形其中的任意两点，则另外一点的可行位置是一段连续区间，且随着两点的顺序移动，剩余一点的可行区间端点是单调的，因此只需要借助余弦定理列出必须满足的不等式并计算即可。由于本题中的 n 比较小，我们可以暴力枚举其中的任意两点，只要在具体实现时稍微注意一下常数即可。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(1)$

1.50 314E-Sereja and Squares (D8805,P5490)

题目大意

对于一个给定的字符串 s，若 s 满足以下所有要求，则 s 被认为是一个合法的字符串：

- 该字符串仅由小写字母和大写字母组成，但是不包括 x 和 X。
- 该字符串的所有字母恰好可以两两配对，且每一对字母必然是同一字母的小写形式与大写形式，同时小写形式的下标必须小于大写形式的下标。
- 每对字母之间确定一条线段，这些线段两两之间只能完全不相交或者完全包含。

现在将长度为 n 的字符串 s 中的所有大写字母以及一部分小写字母擦去，求 s 所对应的合法字符串存在多少种可能的方案。

数据范围

- 20 个测试点的 n 分别为：
- $5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000$；
- $10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000$。

算法分析

由题中的限制条件我们可以发现，其实一个合法的字符串的配对方案是唯一的：因为任意两对字符确定的线段之间不能仅仅相交一部分，所以每个大写字母只能与前面离它最近的一个还未配对的小写字母组合，否则便不合法。基于这一结论，我们完全可以将一个合法的字符串看做一个括号序列，而在这个括号序列中只留下了一部分左括号。现在我们要求可能的合法括号序列有多少种。

因此，若我们将左括号视作 $+1$，将右括号视作 -1，同时记 $dp[i][j]$ 为前 i 位总和为 j 的方案数，则可以得到如下的递推方程：

$$dp[i][j] = \begin{cases} dp[i-1][j-1] & \text{若第 } i \text{ 位已经给定了小写字母} \\ dp[i-1][j-1] \times 25 + dp[i-1][j+1] & \text{若第 } i \text{ 位没有给定字母} \end{cases}$$

显然，$dp[n][0]$ 就是我们所求的答案。不过由于上述递推的时间复杂度为 $\Theta(n^2)$，因此我们需要进行一定的常数优化：

- 对于第 i 位而言，其递推方程中第二维的值上限是 $min(i, n-i)$。
- 对于第 i 位而言，其递推方程中第二维的值一定与 i 奇偶性相同。

通过上述的两个小优化，我们便已经可以在规定时限内通过本题。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n)$

1.51 316D-PE lesson (D9231,P5602)

题目大意

已知有 n 个人，初始时第 i 个人手上有一个编号为 i 的篮球。允许进行若干次交换操作，每次操作可以交换任意两个人手上的篮球。但是，每个人最多只能参与 a_i 次交换操作，并且 a_i 只可能是 1 或者 2。求经过若干次交换操作后不完全相同的持球方案数。

数据范围

- 对于 30% 的数据，$1 \leq n \leq 10$；
- 对于 70% 的数据，$1 \leq n \leq 500$；
- 对于 100% 的数据，$1 \leq n \leq 10^6$。

算法分析

对于本题，不妨通过置换群的形式来考虑。在单个置换中，若不包含或者只包含 1 个只能参与 1 次交换操作的人，则显然可以实现；若包含 2 个，则可以通过相反方向的交换操作来达到目标状态；若包含超过 2 个，则显然无法完成。因此，本题等价于求有多少种不同的置换群，使得每个置换中均只包含至多 2 个只能参与 1 次交换操作的人。

对于转化后的问题，显然答案只与交换次数上限为 1 的人的总数有关。接下来考虑先放置所有交换次数上限为 1 的人，设 $dp[i]$ 表示放置 i 个的方案总数，则不难推出：

$$dp[i] = dp[i-1] + dp[i-2] \times (i-1)$$

上式的前半部分表示单独成为一个置换，而后半部分则表示与另一个人共同构成一个置换。然后，我们考虑将剩余的人加入置换群中，由于新加入的人可以处于任意一个置换中的任意位置，也可以自成一个置换，因此每次新加入一个人后，答案应该乘上 $x+1$，其中 x 表示当前已经处理的总人数。整个算法的时间复杂度为 $\Theta(n)$。

时间复杂度

$\Theta(n)$

空间复杂度

$\Theta(n)$

1.52 316F-Suns and Rays (D9191,P5924)

题目大意

给定一张 $h \times w$ 的位图，已知图中有若干个主体部分呈圆形或椭圆形的太阳，每个太阳连接有若干条宽度为 3 像素的射线。要求识别出所有的太阳以及与每个太阳相连接的射线条数。

数据范围

- $1 \leq h, w \leq 1600$。

算法分析

由于数据保证所有的太阳互不相交，因此只需一遍 BFS 即可得到所有的太阳。接下来主要考虑射线的问题。对于这一问题，我们尝试通过以下一系列的变换将所有的射线相互分离出来，以便于进行计数：

- 首先，我们将所有与背景色相邻的像素也全部都染成背景色，并且连续执行 $3 \sim 4$ 轮这种操作，则由于射线的宽度均为 3 像素，因此所有的射线以及太阳主体部分的边缘均被侵蚀；
- 然后，我们再将所有与前景色相邻的像素全部都染成前景色，并且连续执行 $5 \sim 7$ 轮这种操作，则太阳主体部分的边缘都被还原，但射线依然未被还原，因此所有原图中为前景色而处理过后为背景色的像素就是射线部分，且两两之间相互分离，可以直接统计。

通过上述的分析，我们便只需要对原图进行一些处理后直接计数即可。整个算法的时间复杂度为 $\Theta(hw)$。

时间复杂度

$\Theta(hw)$

空间复杂度

$\Theta(hw)$

1.53 316G-Good Substrings (D8759,P5600)

题目大意

给定字符串 s 和 n 条形如 (p,l,r) 规则，定义字符串 s 满足规则 (p,l,r) 当且仅当：字符串 s 在字符串 p 中的出现次数处于区间 $[l,r]$ 中。求有多少个 s 的子串满足所有的规则。

数据范围

- 对于 30% 的数据，字符串 s 和 p_i 的长度均不超过 200；
- 对于 70% 的数据，字符串 s 和 p_i 的长度均不超过 2000；
- 对于 100% 的数据，字符串 s 和 p_i 的长度均不超过 50000，$0 \leq n \leq 10$。

算法分析

本题显然是一道后缀自动机的模板题，其实也可以借助后缀数组来解决。首先不难发现，当固定子串的左端点为 i 时，则对于任意单个限制规则而言，可行的右端点区间必然是连续的一段。同时，当子串的左端点向右移动时，可行区间的移动必然也是单调的。因此，我们只需要将所有字符串全部拼接在一起，然后对 $height$ 数组构建 ST 表即可。整个算法的时间复杂度为 $\Theta((|s|+\sum|p_i|)\log(|s|+\sum|p_i|))$。

时间复杂度

$\Theta((|s|+\sum|p_i|)\log(|s|+\sum|p_i|))$

空间复杂度

$\Theta(|s|+\sum|p_i|)$

1.54 319D-Have You Ever Heard About the Word (D8803,P5493)

题目大意

给定一个字符串，每次操作删除字符串中最短的一个重复子串，若存在多个重复子串，则选择位置处于最左端的一个。要求不断执行上述操作，直到无法继续执行操作为止，求最后剩下的字符串。

数据范围

- $1 \leq$ 输入字符串长度 ≤ 50000。

算法分析

首先，易知重复子串的长度必然单调不减，因此，只需 $\Theta(n)$ 扫描一遍即可去除所有长度为 l 的重复子串。易证，重复子串的长度显然只会有 $\Theta(\sqrt{n})$ 种可能，因此这一部分的代价为 $\Theta(n\sqrt{n})$。于是，问题转化为判断是否存在长度为 l 的重复子串。

对于这一问题，我们可以每隔 l 个位置取一个断点，则若存在长度为 l 的重复子串，必然也存在两个相邻的断点，使得经过这两个断点的最长公共子串长度至少为 l。因此，我们只需通过二分以及哈希求得 i 和 $i+l$ 的 LCP 与 LCS 之和即可，对于单个长度 l 代价为 $\Theta(\frac{n}{l}\log n)$，总代价为 $\Theta(n\log^2 n)$。整个算法的时间复杂度为 $\Theta(n\log^2 n + n\sqrt{n})$，其中 n 为字符串的长度。

时间复杂度

$\Theta(n\log^2 n + n\sqrt{n})$

空间复杂度

$\Theta(n)$

1.55 319E-Ping-Pong (D9234,P5510)

题目大意

给定 n 个区间，对于任意两个区间 (a,b) 和 (c,d)，可以从 (a,b) 转移到 (c,d) 当且仅当 $c<a<d$ 或者 $c<b<d$。给定 m 次操作，每次操作只可能是加入一个区间，或者查询是否可能从一个已经给定的区间转移到另一个已经给定的区间。保证加入区间的长度单调递增。

数据范围

- 对于 30% 的数据，$1\leq n\leq 10^3$；
- 对于 60% 的数据，$1\leq n\leq 10^4$；
- 对于 100% 的数据，$1\leq n\leq 10^5$，所有输入数均为绝对值不超过 10^9 的整数；
- 数据存在梯度。

算法分析

初看本题可能认为只要直接套用并查集即可，但事实上转移关系并非总是双向的，在有些特殊情况下转移关系是单向的：

- 如果两个区间不存在包含或者被包含关系，则显然两个区间可以相互转移；
- 如果其中一个区间包含另一个区间，则转移只能从小区间向大区间进行。

因此，对于仅有相交关系的两个区间可以直接合并为一个大区间，则剩余的区间只可能有包含和相离两种关系。对于一组询问 x 和 y，若 x 和 y 处于同一个大区间中则显然可以进行转移；若 x 和 y 所处的大区间相离，则显然无法进行转移；否则，只有当 x 区间被 y 所在的大区间包含时才可以进行转移。

对于合并区间的问题，我们可以构建线段树，并将每个区间分解为若干个小区间存储在线段树中的节点上，然后合并时就只需使用并查集即可。整个算法的时间复杂度为 $\Theta(\alpha(n)n\log n)$。

时间复杂度

$\Theta(\alpha(n)n\log n)$

空间复杂度

$\Theta(n\log n)$

1.56 321D-Ciel and Flipboard (D8787,P5521)

题目大意

给定一个 $n \times n$ 的整数矩阵，保证 n 为奇数。每次操作可以将一个 $\frac{n+1}{2} \times \frac{n+1}{2}$ 的子矩阵内的所有数全部乘以 -1。要求通过执行任意次操作，使得矩阵内所有整数之和最大。

数据范围

- 对于 5% 的数据，$n = 1$；
- 对于 35% 的数据，$n \leq 5$；
- 对于 50% 的数据，$n \leq 11$；
- 对于 75% 的数据，$n \leq 15$；
- 对于 100% 的数据，$n \leq 33$。

算法分析

首先，设 $m = \frac{n+1}{2}$，若记每个网格的最终状态为 $st_{i,j}$，其中 $st_{i,j} = 1$ 表示所有操作后网格 (i, j) 中的整数变负，否则 $st_{i,j} = 0$ 表示没有发生改变，则通过观察可以发现：

- $st_{i,j} \oplus st_{m,j} \oplus st_{i+m,j} = 0$；
- $st_{i,j} \oplus st_{i,m} \oplus st_{i,j+m} = 0$。

上述结论的证明显然，因为任意一次操作都只会影响上述等式左侧的任意 0 个或者 2 个值。然后，借助上述结论，我们可以暴力枚举结果矩阵中第 m 行前 m 个整数的最终状态，则剩余的每一行状态都是独立的，可以分别统计最大值。对于每一行，则在枚举了该行第 m 列的状态后，每一列的状态也是相互独立的，因此只需讨论左上角子矩阵中相应位置的状态即可。整个算法的时间复杂度为 $\Theta(2^{\frac{n}{2}}(\frac{n}{2})^2)$。

时间复杂度

$\Theta(2^{\frac{n}{2}}(\frac{n}{2})^2)$

空间复杂度

$\Theta(n^2)$

1.57 323B-Tournament-graph (D8800,P5498)

题目大意

给定 N 个点，要求构造一张竞赛图，使得任意两个节点之间的最短距离都不超过 2。

数据范围

- 对于 40% 的数据，$N \leq 10$；
- 对于 70% 的数据，$N \leq 300$；
- 对于 100% 的数据，$N \leq 1000$。

算法分析

对于一张竞赛图，由于任意两个点之间都有且仅有一条有向边，且题目中要求任意两点间最短距离不超过 2，因此我们容易想到根据奇偶性来构图。下面考虑这样一种构图方式：

- 首先，对于图中的每一个点 i，连接一条有向边 $< i, (i \ mod \ N) + 1 >$；
- 对于剩下的所有点对 $(i, j), i < j$：
  1. 若 i 和 j 奇偶性相同，则连接一条有向边 $< i, j >$；
  2. 若 i 和 j 奇偶性不同，则连接一条有向边 $< j, i >$。

接下来，我们来证明这种构图方式的正确性：首先考虑任意点对 $(i, j), i < j$，且 $i = 1$ 和 $j = N$ 不同时成立，则：

- 若 i 和 j 奇偶性相同，则：
  1. 对于 i 到 j：由构图方式可知，必然存在有向边 $< i, j >$。因此 i 到 j 的最短距离只有 1；
  2. 对于 j 到 i：若 $i \neq 1$，则必然存在路径 $< j, i-1 > \rightarrow < i-1, i >$；若 $j \neq N$，则必然存在路径 $< j, j+1 > \rightarrow < j+1, i >$。因此 j 到 i 的最短距离至多为 2；
- 若 i 和 j 奇偶性不同，则：
  1. 对于 i 到 j：若 $i+1 = j$，则必然存在有向边 $< i, j >$；若 $i+1 \neq j$，则必然存在路径 $< i, j-1 > \rightarrow < j-1, j >$。因此 i 到 j 的最短距离至多为 2；
  2. 对于 j 到 i：若 $i+1 \neq j$，则必然存在有向边 $< j, i >$；若 $i+1 = j$，则当 $N > 4$ 时，路径 $< j, j+2 > \rightarrow < j+2, i >$ 和路径 $< j, i-2 > \rightarrow < i-2, i >$ 必然至少存在一条。因此 j 到 i 的最短距离至多为 2。

在上述的证明中，我们排除了特例 $N = 4$ 的情形，并且通过枚举可知：当 $N = 4$ 时，问题不存在合法解。

最后我们来考虑 $i = 1$ 并且 $j = N$ 的情形：

- 对于 1 到 N：若 N 为奇数，则必然存在路径 $< 1, N-2 > \rightarrow < N-2, N >$；若 N 为偶数，则必然存在路径 $< 1, N-1 > \rightarrow < N-1, N >$。因此 1 到 N 的最短距离至多为 2。

- 对于 N 到 1：由构图方式可知，必然存在有向边 $< N, 1 >$。因此 N 到 1 的最短距离只有 1。

综合上述分析，我们已经证明了该构图方式所构造出的图必然满足题目要求。整个算法的时间复杂度为 $\Theta(N^2)$。

时间复杂度

$\Theta(N^2)$

空间复杂度

$\Theta(1)$

1.58 323C-Two permutations (D8763,P5585)

题目大意

给定两个 $1 \sim n$ 的排列，然后进行 m 次询问，每次询问要求统计出在第一个排列中的位置处于一段给定区间，并且在第二个排列中的位置处于另一段给定区间中的数有多少个。

数据范围

- $1 \leq n \leq 1000000$。
- $1 \leq m \leq 200000$。

算法分析

显然，如果我们用同一个置换作用于这两个排列，则每次询问的答案不会发生改变。因此，对于这两个排列中每个数，我们首先用该数在第一个排列中所处的位置来代替，则每次询问转化成了在第二个排列中查询：一段给定区间中大小在一定范围内的数有多少个。而对于这个问题，我们只需要套用可持久化线段树便可以直接解决。整个算法的时间复杂度为 $\Theta((n+m)\log n)$。

时间复杂度

$\Theta((n+m)\log n)$

空间复杂度

$\Theta(n \log n)$

1.59 325C-Monsters and Diamonds (D8761,P5589)

题目大意

给定 n 只怪物以及 m 条分裂规则，每次操作可以对某一只怪物使用它的某一条分裂规则。保证每次分裂至少会得到一颗钻石，要求将怪物全部分解为钻石。求对于每种怪物可以分解得到的钻石颗数的最大值与最小值。其中，无法彻底完成分解输出 -1，可能分解为无数颗钻石则输出 -2。

数据范围

- $1 \leq m, n \leq 10^5$；
- l_i 的总和不会超过 10^5。

算法分析

对于最小值，显然我们只需使用类似于 $Dijkstra$ 的算法直接计算即可。对于最大值，易知如果一种怪物可以通过若干条分裂规则得到自身，则其对应的最大值为无穷大，否则，剩余的分裂规则中不会存在环，因此只要进行记忆化搜索即可。整个算法的时间复杂度为 $\Theta((n+m)\log n)$。

时间复杂度

$\Theta((n+m)\log n)$

空间复杂度

$\Theta(n+m)$

1.60 325D-Reclamation (D8740,P5682)

题目大意

给定一个底面周长为 c，高度为 r 的圆柱的侧面，要求在任意时刻都必须存在一条从上边界到下边界的四联通路径。每次操作为挖去其中的某个格子，若操作过后仍然满足要求，则执行该操作，否则无视该操作。求成功执行的操作数。

数据范围

- 对于 1% 的数据，$n \leq 5000$；
- 对于 100% 的数据，$r, c \leq 3000$，$n \leq 300000$。

算法分析

首先通过仔细观察可知，若在某一时刻，不存在任意一条从上边界到下边界的四联通路径，则这等价于在所有被挖去的格子中，存在某个八联通的环。有了这一结论，则我们只需判断当前操作是否会使得图中出现环即可。

对于转化过后的问题，我们可以将该圆柱的侧面从任意位置展开，然后将整张图复制一份，即对于所有点 (i, j)，建立其对应点 $(i, j + c)$。借助之前的分析，易知原图中的八联通环等价于新图中的某个点到其对应点的一条八联通路径。因此，我们只需维护每个点与其对应点的连通性即可，而这一问题借助并查集即可实现。整个算法的时间复杂度为 $\Theta(n\alpha(rc))$。

时间复杂度

$\Theta(n\alpha(rc))$

空间复杂度

$\Theta(rc)$

1.61 325E-The Red Button (D9224,P5730)

题目大意

已知初始时权值为 0，每次操作可以将当前权值 x 变为 $2x \ mod \ n$ 或者 $(2x+1) \ mod \ n$。要求在执行 n 次操作后恰好遍历每个数恰好一次，并且要求最终回到值为 0。求任意一组可行方案。

数据范围

- 对于 15% 的数据，$2 \leq n \leq 10$；
- 对于 30% 的数据，$2 \leq n \leq 20$；
- 对于 100% 的数据，$2 \leq n \leq 10^5$。

算法分析

首先易证，当 n 为奇数时显然无解，因此下面只考虑 n 为偶数的情形。其次，不难发现对于每个非零数，必然只会恰好存在两条入边以及两条出边。同时，设 $m = \frac{n}{2}$，则 x 和 $x + m$ 的出边完全一样，并且可以交叉互换，因此我们可以使用一种类似于求欧拉回路的算法，即：首先访问 $2x \ mod \ n$，若在回溯之后 $(2x + 1) \ mod \ n$ 仍然未被访问，则继续访问 $(2x + 1) \ mod \ n$，遍历结束之后只需倒序输出弹栈序列即可。整个算法的时间复杂度为 $\Theta(n)$。

时间复杂度

$\Theta(n)$

空间复杂度

$\Theta(n)$

1.62 329D-The Evil Temple and the Moving Rocks (D8775,P5550)

题目大意

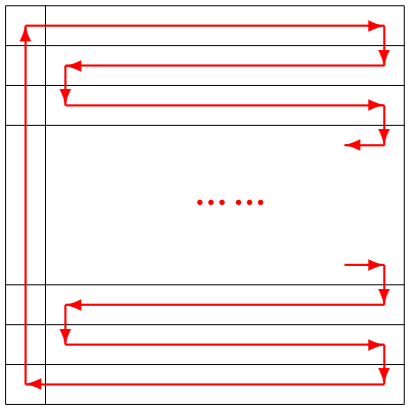
在一张 $n \times n$ 的网格上，要求在其中某一些网格上放置某种传动装置。这些传动装置在被激活后，将会沿着给定的方向（上、下、左、右）一直移动，直到撞到边界或其他传动装置后停止移动。撞击过后，该传动装置停止运动，被撞击到的传动装置将会被激活。当所有传动装置被激活的总次数达到 10^7 或者传动装置撞到边界后，整个过程结束。在每次撞击中，若传动装置在撞击发生之前至少移动了一步，则该次撞击会有 1 的贡献。题目给定 x，要求设计一种传动装置的初始摆放方案，并给定最开始激活的传动装置所处的位置，使得在整个过程结束之前，至少有 x 的贡献。

数据范围

- $n \leq 300$，数据有梯度。
- 对于 30% 的数据，$x \leq k$。
- 对于 60% 的数据，$x \leq 2k^2 - 2k$。
- 对于 100% 的数据，$x \leq k^3 - k^2$。
- 数据保证有解。

算法分析

题目中所要求的总贡献约为 $(\frac{n}{2})^3$，因此我们已经不能单纯地依靠仅仅一次对整个网格的遍历来达成任务。考虑如下图中的一种可行的循环传动路径：



在该方案中，位于最左侧的第一列被用来连接第一行与最后一行，以使得整个传动过程可以循环进行。因此，第一列除了第一格外的其他所有格中全部放置向上的传动装置，第一格则放置向右的传动装置。对于网格中剩下的部分，我们考虑使用类似于之前的放置方案：

首先，对于除了最后一行外的其他所有行，每行均有且仅有一个向下的传动装置，并且在奇数行放置在最后一列，在所有偶数行放置在第二列。

然后，为了保证整个传动过程的持续性，在每一轮的循环传动中，我们都必须保证每个向下的传动装置下方都存在一个传动装置，否则整个传动路线将被破坏。因此，在每一行的起始部分（奇数行的起始部分是从第二列向右，偶数行的起始部分是从最后一列向左）都必须连续放置足够多的同向传动装置，以保证在总贡献达到要求之前，每一轮传动在经过该处时都能正常进行。

接下来，考虑对于每一行，我们假定起始部分连续放置的传动装置有 p 个，则剩余的部分长度为 $q = n - 2 - p$。在剩余的部分，我们将所有网格交错地设定为空地与传动装置。

- 当 q 为奇数时，若我们在剩余部分放置 $\frac{q-1}{2}$ 个传动装置，则每一行在每一轮的传动中将会有 $\frac{q+1}{2}$ 的贡献，且传动总共可以进行 p 轮。通过数学计算可知，当 $p = k - 1$ 且 $q = k - 1$ 时，所有行的总贡献达到最大值 $\frac{p(q+1)}{2} \times n = \frac{1}{2}(k-1)k \cdot 2k = k^3 - k^2$。
- 当 q 为偶数时，若我们在剩余部分放置 $\frac{q}{2}$ 个传动装置，则每一行在每一轮的传动中将会有 $\frac{q}{2}$ 的贡献，且传动总共可以进行 $p + 1$ 轮。通过数学计算可知，当 $p = k - 1$ 且 $q = k - 1$ 时，所有行的总贡献达到最大值 $\frac{q(p+1)}{2} \times n = \frac{1}{2}(k-1)k \cdot 2k = k^3 - k^2$。

因此，综合以上分析，我们只需要分奇偶讨论，并且对于每一行依次处理即可构造出合法的方案，同时选取 $(1, 1)$ 作为传动的起始点，即可顺利解决本题。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n^2)$

1.63 329E-Evil (D9280,P5661)

题目大意

给定平面直角坐标系中的 n 个点，定义任意两点之间的距离为两者间的曼哈顿距离。求哈密尔顿回路的最大可能长度。

数据范围

- 对于 30% 的数据，$n \leq 10$；
- 对于 100% 的数据，$3 \leq n \leq 10^5$，$0 \leq x_i, y_i \leq 10^9$。

算法分析

对于本题，一个大致的思路是分别按照横纵坐标将整个二维平面近似地等分成四个部分，且任意两个不同部分中的点数之差至多为 1，则答案的上限必然是对于横纵坐标分别用较大的一半减去较小的一半。至于上界是否可达，则通过一些分类讨论可知，该上限并非总是可以取到，但对于无法取到上限的情形，则我们总是可以通过只执行一次局部调整来取到次大的上限值。整个算法的时间复杂度为 $\Theta(n \log n)$。

时间复杂度

$\Theta(n \log n)$

空间复杂度

$\Theta(n)$

1.64 331C-The Great Julya Calendar (D8755,P5621)

题目大意

给定一个正整数 n，每次操作可以将 n 减去它某个数位上的数字。求最少需要操作多少次，才能将 n 减到 0。

数据范围

- $0 \leq n \leq 10^{18}$。

算法分析

对于这个问题，我们首先可以证明下面的贪心是正确的：要使用最少的操作次数，则我们每次应该减去 n 中最大的数字。要证明这一结论的正确性，我们相当于要证明随着 n 的递增，其所需要的操作次数单调不减。

对于转化后的结论，我们可以有如下的归纳证明：

- 若 n 的最后一位数字不是 9，则 n 和 $n+1$ 仅有最后一位数字不同。因此，对于 $n+1$ 的最优策略：
  1. 如果第一步减去的数字不是最后一位上的数字，则对于 n 的策略，第一步也可以减去相同位置上的数字，两者的大小关系不变。因此 n 所需要的操作次数不多于 $n+1$ 所需要的操作次数；
  2. 如果第一步减去的数字就是最后一位上的数字，则对于 n 的策略，第一步也可以减去最后一位上的数字，两者的值变成相等的。因此 n 所需要的操作次数不多于 $n+1$ 所需要的操作次数；
- 若 n 的最后一位数字就是 9，假设 n 的末尾有连续 k 个 9，且其倒数第 $k+1$ 位的数字为 p，则 $n+1$ 的末尾有连续 k 个 0，且其倒数第 $k+1$ 位的数字为 $p+1$。因此，对于 $n+1$ 的最优策略：
  1. 如果第一步减去的数字不是最后 $k+1$ 位的数字，则对于 n 的策略，第一步也可以减去相同位置上的数字，两者的大小关系不变。因此 n 所需要的操作次数不多于 $n+1$ 所需要的操作次数；
  2. 如果第一步减去的数字就是倒数第 $k+1$ 位的数字 $p+1$，则对于 n 的策略，第一步也可以减去倒数第 $k+1$ 位上的数字 p，两者的值变成相等的。因此 n 所需要的操作次数不多于 $n+1$ 所需要的操作次数。

借助上面的结论，我们就可以按位进行记忆化搜索：记当前剩余的数字为 rem，且更高的数位上提供的最大数字为 mx，同时用 $dp[rem][mx]$ 来表示在这种状态下，将 rem 减为 0 所需的最小步数，然后每次将当前 rem 的最高位取出，递归进行 DP 直到 rem 变为个位数为止。整个算法的时间复杂度为 $\Theta(\log n)$。

时间复杂度

$\Theta(\log n)$

空间复杂度

$\Theta(\log n)$

1.65 332D-Theft of Blueprints (D9212,P5764)

题目大意

给定一张 n 个点的带权无向图，保证对于任意一个大小为 k 的顶点集合 S，有且仅有一个点 $v(S)$ 与 S 中的所有点都有边相连，操作的代价为 $v(S)$ 与 S 中的所有点连边的边权和。求进行一次操作的期望代价。

数据范围

- $1 \leq k < n \leq 2000$；
- $-1 \leq c_{i,j} \leq 10^9$。

算法分析

通过分类讨论可以证明，满足题中所有限制的带权无向图只可能有如下三种情形：

- $k = 1$，则整张图仅由若干条不相交的边构成；
- $k = 2$，则整张图由共用同一个中心点的若干个三元环组成；
- $n = k + 1$ 且整张图为一张完全图。

借助上述结论，则只需分类统计每条边的贡献即可。其实，由于本题点数较少，即使没有得出上述结论也无大碍，可以直接利用组合数暴力统计每条边的贡献。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n^2)$

1.66 333C-Lucky Tickets (D9199,P5901)

题目大意

给定一个正整数 k，定义一个允许有前导零的 8 位十进制数为幸运数，当且仅当：存在某种方案，使得在该数字的某些位置之间插入若干个运算符号（包括‘+’、‘-’、‘*’）以及括号之后，表达式的运算结果等于 k。要求构造出 m 个不同的幸运数。

数据范围

- $0 \leq k \leq 10^4$，$1 \leq m \leq 3 \times 10^5$。

算法分析

通过打表可以发现，本题合法解的数量其实非常之多，因此不妨直接暴力枚举所有 4 位数的所有可能表达式。然后，不妨通过将两个 4 位数进行拼接来构造答案，只需保证高 4 位和低 4 位可以通过加减乘中的任意一种运算得到 k 即可。整个算法的时间复杂度为 $\Theta(m)$。

时间复杂度

$\Theta(m)$

空间复杂度

$\Theta(m)$

1.67 338D-GCD Table (D8751,P5628)

题目大意

对于一张 $n \times m$ 的表格，定义其中第 i 行第 j 列的数为 $gcd(i,j)$。现在给定一个长度为 k 的整数序列 a_i，询问整个序列是否在表格中某一行的连续位置出现过。

数据范围

- $n, m \leq 10^{12}$，$k \leq 10000$，数列中的每一个数 a_i 都不超过 10^{12}。

算法分析

在本题中要求整个序列出现在表格的某一行中，因此首先尝试确定行的位置。首先，若该序列出现在第 i 行，则容易得到 $lcm(a_i) \mid i$。下面证明，若存在合法的方案，则 $i = lcm(a_i)$ 必定是其中一解。

- 记 $A = lcm(a_i)$，假设 $i = A$ 不是合法解，并且当 $i = x \times A$ 时，存在一组合法的方案；
- 易知，在这种情况下，必然存在某一列 j，使得 $gcd(A, j) \neq gcd(xA, j)$。可以推知，必然存在某一质因子的幂 p^q，使得 $p^q \mid xA$ 且 $p^q \mid j$，但 $p^q \mid A$ 不成立；
- 然而，由于 $A = lcm(a_i)$，且存在某一值 $a_t = gcd(xA, j)$。同时，又因为 $p^q \mid a_t$，所以 $p^q \mid A$ 必然成立，与假设矛盾。

通过上述分析，我们可以方便地求得可能的行号 i。接下来，我们尝试确定列的初始位置。

- 假设可能的初始列号为 j，则根据题目条件，我们可以列出下列同余方程组：

$$\begin{cases} j \equiv 0 \pmod{a_1} \\ j \equiv -1 \pmod{a_2} \\ j \equiv -2 \pmod{a_3} \\ \vdots \quad \vdots \quad \vdots \\ j \equiv -i \pmod{a_{i+1}} \\ \vdots \quad \vdots \quad \vdots \\ j \equiv -(k-1) \pmod{a_k} \end{cases}$$

- 易知，可能的初始列号 j 必然满足上述同余方程组。同时，由中国剩余定理可知，若上述同余方程组有解，则所有解在模 A 的意义下是唯一的。另外，又因为行号 i 就是 A，所以 $gcd(i, j + t \times A) = gcd(i, j)$，即上述同余方程组的所有解都等价。因此，$j$ 只需取模 A 意义下的解即可；
- 对于上述同余方程组，只需借助扩展欧几里德算法两两合并即可。

在求解过程中，若行（列）的位置超过限制，则可以直接结束算法。否则，只需在最后再进行验证即可。整个算法时间复杂度为 $\Theta(k \log A)$。

时间复杂度

$\Theta(k \log A)$

空间复杂度

$\Theta(k)$

1.68 338E-Optimize! (D9166,P6043)

题目大意

给定以下一段代码，要求对算法进行等价优化，实现完全相同的功能。

```
getAnswer(a[1..n], b[1..len], h)
  answer = 0
  for i = 1 to n-len+1
    answer = answer + f(a[i..i+len-1], b, h, 1)
  return answer

f(s[1..len], b[1..len], h, index)
  if index = len+1 then
    return 1
  for i = 1 to len
    if s[index] + b[i] >= h
      mem = b[i]
      b[i] = 0
      res = f(s, b, h, index + 1)
      b[i] = mem
      if res > 0
        return 1
  return 0
```

数据范围

- 对于 10% 的数据，$1 \leq len \leq n \leq 10$；
- 对于 40% 的数据，$1 \leq len \leq n \leq 1000$；
- 对于 100% 的数据，$1 \leq len \leq n \leq 150000$；
- 对于 100% 的数据，$1 \leq a_i, b_i, h \leq 10^9$。

算法分析

首先通过观察给定的代码可知，所求的即为：序列 a 有多少个长度为 m 的子串能与序列 b 匹配。其中，两个数 x 和 y 能够匹配当且仅当 $x + y \geq h$，两个序列能够匹配则当且仅当存在完备匹配。

易知，两个序列的匹配可以在排完序后直接贪心进行，即对于任意 b_i 必须存在至少 $m - i + 1$ 个能够与之匹配的数。对于这一问题，显然可以借助线段树来解决，则所用到的操作只有区间修改和查询区间最小值。整个算法的时间复杂度为 $\Theta(n \log m)$。

时间复杂度

$\Theta(n \log m)$

空间复杂度

$\Theta(n + m)$

1.69 339E-Three Swaps (D8768,P5580)

题目大意

给定一个 $1 \sim n$ 的排列，每次操作可以任选一段区间 $[l, r]$，并将该段区间内的序列整体翻转。要求通过至多三步操作将序列还原为 $1 \sim n$。数据保证在三步操作之内必然有解。

数据范围

- $1 < n \leq 1000$。

算法分析

对于前两步操作而言，显然操作的区间 $[l, r]$ 必然满足：序列在翻转后至少可以接上一处“断层”，即满足 $|a_l - a_{r+1}| = 1$ 或者 $|a_{l-1} - a_r| = 1$。对于第三步操作，虽然可能不满足上述约束条件，但通过若干分类讨论可以发现，即使在特殊情况下，同样必然存在一种三步操作均满足约束条件的操作序列，因此只需直接进行搜索即可。整个算法的时间复杂度为 $\Theta(n^2)$。

时间复杂度

$\Theta(n^2)$

空间复杂度

$\Theta(n)$

1.70 341E-Candies Game (D9221,P5733)

题目大意

给定 n 个箱子，初始时每个箱子内都有数量已知的糖果。每次操作可以选择任意两个不同的箱子 i 和 j，不妨设 $a_i \leq a_j$，则该次操作将从箱子 j 中取出 a_i 个糖果放入箱子 i 中。要求通过不超过 10^6 步操作，使得恰好只有两个箱子中有糖果。求任意一种合法操作序列。

数据范围

- 对于 15% 的数据，$n = 3$；
- 对于 30% 的数据，$n \leq 10$，$a_i \leq 10$；
- 对于 100% 的数据，$n \leq 10^3$，$\sum a_i \leq 10^6$。

算法分析

显然本题是一道构造题，考虑对于任意三个非空的箱子 i、j 和 k，不妨设 $a_i \leq a_j \leq a_k$，则我们尝试证明，在 $\Theta(\log^2 a)$ 步操作之内必然可以将其中至少一个箱子变为空。证明如下：

- 设 $a_j = xa_i + u$，$a_k = ya_i + v$，x 的二进制表示为 $(x_p x_{p-1} \dots x_1 x_0)_2$，$y$ 的二进制表示为 $(y_q y_{q-1} \dots y_1 y_0)_2$；
- 对于 a_i，考虑依次执行 c 步操作，对于第 w 步操作，若 $x_w = 1$ 则执行操作 (i, j)，否则执行操作 (i, k)；
- 由于 $x \leq y$，则上述的每步操作都必然可以执行；同时，在上述的一系列操作结束之后，必有 $a_j = u$，即 $a_j = a_j \ mod \ a_i$，因此在 $\Theta(\log a)$ 步操作之后，三个数中的最小值至少减小了一半；
- 由归纳可知，在执行 $\Theta(\log a)$ 轮系列操作之后，三个数中的最小值必然为 0。

通过以上分析可知，对于任意三个箱子都可以在 $\Theta(\log^2 a)$ 步操作之内清空其中至少一个箱子，因此除非初始时有糖果的箱子数小于 2，否则必然存在一组合法解。整个算法的时间复杂度为 $\Theta(n \log^2 a)$。

时间复杂度

$\Theta(n \log^2 a)$

空间复杂度

$\Theta(n)$

1.71 343E-Pumping Stations (D9219,P5735)

题目大意

给定一张 n 个点 m 条边的带权无向图，要求构造出一个排列，其对应的权值为排列中任意相邻两点之间在原图中的最小割的权值之和。求可能的最大权值，并且要求给出相应的方案。

数据范围

- 对于 40% 的数据，$n \leq 30$，$m \leq 350$；
- 对于 60% 的数据，$n \leq 80$，$m \leq 1000$；
- 对于 100% 的数据，$n \leq 200$，$m \leq 1000$，$1 \leq c_i \leq 100$。

算法分析

本题是一道 *Gomory–Hu tree* 的模板题，具体资料可以参见维基百科。根据给出的定理以及相关基本性质，我们只需执行 $\Theta(n)$ 次网络流算法即可构造出 *Gomory–Hu tree*。然后，我们每次只需找到树上权值最小的树边，并对分割出来的两棵子树递归构造出答案即可。整个算法的时间复杂度为 $\Theta(n^3m)$。

时间复杂度

$\Theta(n^3m)$

空间复杂度

$\Theta(n+m)$

1.72 346E-Doodle Jump (D9222,P5732)

题目大意

给定数轴上的 n 个点，其中第 i 个点的坐标为 $ai \ mod \ p$，数据保证 a 和 p 互质。初始时处于坐标原点，每一步可以跳跃到距离不超过 h 的点上。询问是否可能通过若干次跳跃到达最右端的点上。

数据范围

- 对于 30% 的数据，有 $t \leq 1000$，$n \leq 1000$；
- 对于 100% 的数据，有 $1 \leq t \leq 10^4$，$1 \leq a \leq 10^9$，$1 \leq n < p \leq 10^9$，$0 \leq h \leq 10^9$。

算法分析

首先，我们考虑将 $[0, p)$ 按照长度为 a 进行分段，则除去最后不完整的一段，剩余的所有段至多只有两种不同的情形，且两者的差别只可能有一个数。其次，本题所求其实就是相邻两点间距离的最大值，而通过分类讨论显然可以发现最大值不可能出现在最后的不完整段中。同时，对于所有的完整段而言，则最大值必然出现在整数个数较少的段中。因此，我们可以直接用区间 $[(\lfloor \frac{p}{a} \rfloor - 1)a, \lfloor \frac{p}{a} \rfloor a)$ 内的情形来代替。此时，唯一的特例是 $an < p$ 的情形，否则，我们可以将 p 和 a 分别用 a 和 $a - p \ mod \ a$ 来代替。于是，我们可以通过上述方法来缩小范围。但是，如果只是借助以上变换，算法的总代价依然没有保证，不过由于 $p \ mod \ a$ 和 $a - p \ mod \ a$ 的情形完全等价，因此我们每次只需选取其中较小的一个即可。整个算法的时间复杂度为 $\Theta(t \log a)$。

时间复杂度

$\Theta(t \log a)$

空间复杂度

$\Theta(1)$

1.73 351D-Jeff and Removing Periods (D8798,P5500)

题目大意

对于一个给定序列，每次操作可以任意选择一个下标成等差数列，且权值均相等的子序列，然后删去被选中的所有数，同时任意重新排列剩余的数。要求通过最少的操作次数将所有数删去。现在给定一个长度为 m 的序列，进行 q 次询问，每次查询将 l_i 到 r_i 的子串作为初始串时，所需的最小操作次数。

数据范围

- 对于 30% 的数据，$m, q \leq 2000$；
- 对于 100% 的数据，$1 \leq m, q, b[i] \leq 10^5$，$1 \leq l \leq r \leq m$。

算法分析

通过观察，我们首先可以发现：在经过第一次操作之后，我们可以将剩余的所有数从小到大排列，之后所有相同的数便均可以通过一次操作全部删去。因此每一次询问可以等价地转化为以下两个询问：

- 查询区间 l_i 到 r_i 中有多少种不同的数；
- 查询区间 l_i 到 r_i 中，是否存在某个数可以只用一次就全部删去，即在 l_i 到 r_i 中，该数出现的所有下标成等差数列。

对于上述的两类查询，我们均可以借助莫队算法来完成：

- 对于第一类查询，只需维护计数器即可；
- 对于第二类查询，我们可以计算出每个数与左（右）侧离它最近的一个相同的数的下标差，然后通过哈希表统计出每种下标差出现的次数，以及每个数出现的下标差的种数。这样便可以快速地维护有几种数可以只用一次全部删去。

整个算法的时间复杂度为 $\Theta((m+q)\sqrt{m})$。

时间复杂度

$\Theta((m+q)\sqrt{m})$

空间复杂度

$\Theta(m)$

1.74 354D-Transferring Pyramid (D9192,P5918)

题目大意

给定一个高度为 n 的金字塔，要求仅通过使用两种给出的操作，使得给定的 k 个位置需要至少被修改一次，其他位置无要求，求最小总代价。其中，给定的两种操作分别为：

- 修改金字塔中的某个位置，单次操作的代价为 3；
- 修改金字塔中以某个位置为顶点的小金字塔中的所有位置，单次操作的代价为小金字塔中位置总数加二。

数据范围

- 对于 30% 的数据，$1 \leq n, k \leq 200$；
- 对于 60% 的数据，$1 \leq n, k \leq 2000$；
- 对于 100% 的数据，$1 \leq n, k \leq 10^5$。

算法分析

首先，不妨将整个金字塔逆时针旋转 45°，则整个金字塔被转化为一个上三角，即第 i 行有 $n-i+1$ 个元素。其次，易证每行每列都至多只会选择一个位置执行操作二，且随着行号的递增，所选位置的列号必然递减。

因此，考虑设计 DP 的状态为 $dp[i][j]$，其中 i 和 j 分别表示行号和列号，状态对应的值则表示在满足前 i 行所有位置的要求，且当前行选择了第 j 列执行操作二的情形下最小的总代价。然后，根据操作二位置的单调性可知，当前行第 j 列的状态只会由上一行右侧的某一列转移而来，因此整个 DP 过程的总代价为 $\Theta(n^2)$。最后，由于选择某一行倒数第 j 列执行操作二的代价为 $\frac{j(j+1)}{2}+2$，而即使对所有存在要求的位置都执行操作一的总代价也只有 $3k$，因此 j 的取值范围只会达到 $\Theta(\sqrt{k})$ 级别。整个算法的时间复杂度为 $\Theta(n\sqrt{k})$。

时间复杂度

$\Theta(n\sqrt{k})$

空间复杂度

$\Theta(\sqrt{k})$

Chapter 2

Google Code Jam

2.1 WF2008E-The Year of Code Jam (D9207,P5787)

题目大意

给定一张 $n \times m$ 的 01 网格，对于每个网格有三种属性：

- 必须填 0；
- 必须填 1；
- 随意填 0 或者填 1。

对于所有填 1 的网格，其收益为 4 – 与之相邻的网格中 0 的个数。求最大总收益。

数据范围

- 对于 10% 的数据，$n, m \leq 5$；
- 对于 30% 的数据，$n, m \leq 15$；
- 另 20% 的数据，$m \leq 10$；
- 另 20% 的数据，$T = 1$；
- 对于 100% 的数据，$n, m \leq 50$，$T \leq 100$。

算法分析

首先，不妨将总收益的定义等价地修改为：相邻的两个网格内若数值相异则贡献为 1，否则贡献为 0。由于题目中要求最大化总收益，但是这样不利于建立模型，因此不妨对整个网格进行黑白染色，然后将所有白色网格内的数值取反，则原问题等价于：相邻的两个网格内若数值相异则代价为 1，否则代价为 0，要求最小化总代价。对于这一问题，显然我们可以构建最小割模型来解决。整个算法的时间复杂度为 $\Theta(Tn^3m^3)$。

时间复杂度

$\Theta(Tn^3m^3)$

空间复杂度

$\Theta(nm)$

2.2 WF2009A-Year of More Code Jam (D8786,P5522)

题目大意

给定 T 种比赛，已知第 i 种比赛共有 m_i 轮，且第 j 轮比赛会在比赛开始后的第 $d_{i,j}$ 天举行。现已知所有 T 种比赛会随机地在 n 天内的某一天开始，求 $\sum_{i=1}^{n} s_i^2$ 的期望，其中 s_i 表示第 i 天举行比赛的场数。要求以带分数形式输出。

数据范围

- 对于 100% 的数据，$1 \leq n \leq 10^9$，$2 \leq m \leq 50$，$1 < d[2] < d[3] < \ldots < d[m] \leq 10000$，$T \leq 50$。

算法分析

对于这一问题，若设 $x_{i,j}$ 表示第 i 天第 j 种比赛举办一轮比赛的概率，则由于期望的性质可知：

$$\sum_{i=1}^{n} s_i^2 = \sum_{i=1}^{n} (\sum_{j=1}^{T} x_{i,j})^2$$

易知，第 i 天第 j 种比赛举办一轮比赛的概率 $x_{i,j} = \frac{k}{n}$，其中 k 表示 $d_{j,1}, d_{j,2}, \ldots, d_{j,m_j}$ 中不超过 i 的数的个数。同时，由于 d 的值不超过 10000，而且当 i 增大时概率不会再发生改变，因此，我们只需要在 DP 的过程中维护带分数的形式即可。整个算法的时间复杂度为 $\Theta(DT)$。

时间复杂度

$\Theta(DT)$

空间复杂度

$\Theta(DT)$

2.3 WF2009B-Min Perimeter (D9294,P6105)

题目大意

给定平面直角坐标系中的 n 个整点，求仅使用这些点构成的所有三角形中的最大周长（允许退化的三角形，即三点共线的情形）。

数据范围

- $0 < n \leq 100000$。

算法分析

对于本题而言，我们可以对经典的平面最近点对算法进行一些修改，进而解决本题。考虑将当前点集按照横坐标分成均匀的两部分，则对于所有三个顶点处于同一侧的情形只需递归处理即可。下面着重考虑三个顶点分居中轴线两侧的情形：

- 假设在左右两侧递归均执行完毕后，当前的全局最优解为 p；
- 显然，所有距离中轴线的距离超过 $\frac{p}{2}$ 的点必然不可能出现在最优解上，设左右两侧剩余的点集分别为 L 和 R；
- 由于三个顶点分居中轴线两侧，因此必然是在两侧点集中分别选取出 1 个和 2 个点，不妨以在 L 中选取 1 个点，并在 R 中选取 2 个点的情形为例；
- 易知，R 中取出的两个点与 L 中枚举的点纵坐标之差也不超过 $\frac{p}{2}$，则我们将区域的范围缩小到了 $\frac{p}{2} \times p$；
- 可以证明，在 $\frac{p}{2} \times p$ 的区域中，总点数必然不超过 16，因此暴力枚举这些点的总代价可以承受。

通过上述分析，我们只需在平面最近点对算法的基础上进行若干改动即可。整个算法的时间复杂度为 $\Theta(n \log n)$。

时间复杂度

$\Theta(n \log n)$

空间复杂度

$\Theta(n)$

2.4 WF2009C-Doubly-sorted Grid (D8789,P5519)

题目大意

给定一张 $R \times C$ 的网格，其中某些已知网格中已经写有某个小写字母，其余网格为空。要求在所有空网格中均填入一个小写字母，使得每一行的小写字母从左到右单调不下降，且每一列的小写字母从上到下单调不下降。求可行的方案总数。

数据范围

- $1 \leq T \leq 3$，$1 \leq R, C \leq 10$。
- 网格中每个字符都是'.'或一个小写英文字母。
- 对于 $MAX(R, C) = 10$ 的数据，$T \leq 2$。

算法分析

首先可以发现，网格中字典序不超过某个字符的所有小写字母填充了整个网格的左上角，并且构成的右边界单调不增。因此，考虑以分界线为状态进行 DP。由于分界线只会向上或向右延伸，因此状态总数只有 C_{R+C}^{R} 种。

下面考虑以 $DP[st][i]$ 为状态，表示分界线状态为 st 并且只使用了前 i 种字符的方案数，则状态的转移有以下两种情形：

- 没有第 i 种字符，则相应的值为 $DP[st][i-1]$；
- 若有第 i 种字符，则可以发现，对于分界线的所有转角处，第 i 种字符至少出现在其中一处位置。

对于第二种转移方式，由于重复计数的问题，我们可以借助容斥原理来解决即可。整个算法的时间复杂度为 $\Theta(|S|C_{R+C}^{R}2^{min(R,C)})$，其中 S 表示字符集。

时间复杂度

$\Theta(|S|C_{R+C}^{R}2^{min(R,C)})$

空间复杂度

$\Theta(|S|C_{R+C}^{R})$

2.5 WF2009D-Wi-fi Towers (D9232,P5548)

题目大意

给定 n 个通信站，每个通信站都有一定的覆盖范围。当前所有通信站均使用 A 类协议，将某个通信站升级为 B 类协议有一定的收益（负收益表示成本）。同时，要求所有处于某个使用 B 类协议的通信站的覆盖范围内的通信站也必须使用 B 类协议。要求最大化总收益。

数据范围

- 对于 40% 的数据，$n \leq 15$；
- 对于另外 20% 的数据，保证所有 x 或者所有 y 都相同；
- 对于 100% 的数据，$n \leq 500$，$1 \leq T \leq 55$，$-10000 \leq x, y \leq 10000$，$1 \leq r \leq 20000$，$-1000 \leq s \leq 1000$；
- 没有两个塔有相同的坐标；
- 数据存在梯度。

算法分析

本题是一道比较裸的最大权闭合子图问题，考虑构建最小割模型，只需要从源点连向所有正收益的通信站，并且从所有负收益的通信站连向汇点即可。对于通信站之间的限制条件，则只需借助容量为正无穷的边即可。整个算法的时间复杂度为 $\Theta(Tn^4)$。

时间复杂度

$\Theta(Tn^4)$

空间复杂度

$\Theta(n^2)$

2.6 WF2010A-Letter Stamper (D9177,P5984)

题目大意

给定一个仅由 ABC 三种字符构成的字符串 S。初始时栈为空，每次操作可以为以下三种操作中的任意一种：

- 压入一个字符到栈顶；
- 弹出当前的栈顶字符；
- 打印当前的栈顶字符。

要求结束所有操作后栈依然为空。求打印出给定字符串的最少操作次数。

数据范围

- 40% 的数据，$T = 1$，$|S| \leq 100$；
- 100% 的数据，$T \leq 20$，$\sum |S| \leq 2000$。

算法分析

通过一些观察可以发现以下的若干性质：

- 若当前栈顶字符即为当前所需打印的字符，则必然立即进行打印操作；
- 栈中任意相邻的两个字符必然互不相同；
- 栈中必然不可能出现形如 ABA 的连续字符，因为在这种情形下，我们总是可以用弹栈操作来代替压栈操作。

借助上述所有结论可知，在最优方案中，栈中元素必然是每三个一循环。因此，我们可以直接在 DP 状态中顺带记录下栈中前三个元素即可，则转移方程显然。整个算法的时间复杂度为 $\Theta(|S|^2)$。

时间复杂度

$\Theta(|S|^2)$

空间复杂度

$\Theta(|S|)$

2.7 WF2010C-Candy Store (D8794,P5511)

题目大意

已知有 k 个人，每个人可以选择任意一个 $1 \sim C$ 的正整数。要求预先选择一个可重复的正整数集合，使得必然可以依次满足每个人的要求。其中，定义满足某个人的要求当且仅当：可以从当前剩余的正整数集合中选取出若干个正整数，使得这些正整数之和恰好为其初始时所选择的正整数。

数据范围

- 对于 25% 的数据，$k \leq 20$，$C \leq 3$；
- 对于另外 20% 的数据，$k \leq 1000$，$C \leq 10$；
- 对于另外 5% 的数据，$k = 1$；
- 对于 100% 的数据，$1 \leq T \leq 100$，$1 \leq k \leq 1000$，$1 \leq C \leq 10^{12}$。

算法分析

对于本题，显然我们需要找到一种具有通用性的构造算法。假设选取的正整数从小到大依次为 $a_1, a_2, \ldots, a_n$，则我们尝试证明最优方案一定满足：对于任意 $1 \leq i \leq n$，必有 $a_i = \lfloor \frac{1}{n} \sum_{j=1}^{i-1} a_j \rfloor + 1$。下面证明该方案的最优性如下：

- 首先证明其充分性。假设存在某个 i，使得 $a_i > \lfloor \frac{1}{n} \sum_{j=1}^{i-1} a_j \rfloor + 1$，其等价于 $n(a_i - 1) > \sum_{j=1}^{i-1} a_j$。此时，若每个人所选取的正整数均为 $a_i - 1$，则该方案显然不合法，因此必有 $a_i \leq \lfloor \frac{1}{n} \sum_{j=1}^{i-1} a_j \rfloor + 1$；
- 接下来证明必要性。利用数学归纳法可知，若我们在每次处理需求时，直接贪心地用最小的满足条件的数来满足需求，则剩余的所有数依然满足该性质。

通过上述分析，我们便可以直接构造出最优的方案。整个算法的时间复杂度为 $\Theta(T \log C)$。

时间复杂度

$\Theta(T \log C)$

空间复杂度

$\Theta(1)$

2.8 WF2011A-Runs (D9229,P5626)

题目大意

给定一个字符串，定义其权值为由同一字符构成的极大子串的个数。求有多少种重新排列给定字符的方案，使得其权值不发生改变。

数据范围

- 对于 50% 的数据，字符串长度不超过 100；
- 对于 100% 的数据，字符串长度不超过 450000，run 不超过 100 个。

算法分析

对于这一问题，显然可以通过 DP 来解决。不妨设计 DP 状态为 $dp[i][j]$，表示将所有前 i 种字符重新排列，并且恰好形成 j 段的方案总数。考虑插入所有第 $i+1$ 种字符，若共分成 k 段插入，则有以下两种可能的情形：

- 若某一段字符插在另一段连续字符的内部，则总段数增加 2；
- 若某一段字符插在了原本的字符段相接处，则总段数增加 1。

由于本题中段数较少，因此我们可以直接暴力枚举上述两种情形的段数，然后进行转移即可。整个算法的时间复杂度为 $\Theta(n^3|S|)$，其中 S 表示字符集，n 表示原始字符串的权值。

时间复杂度

$\Theta(n^3|S|)$

空间复杂度

$\Theta(n|S|)$

2.9 WF2011B-Rains Over Atlantis (D8747,P5632)

题目大意

给定一张 $H \times W$ 的地图，给定每个网格的初始非负海拔，地图外的海拔为负无穷。对于每次操作，首先使得所有网格内积累尽可能多的水而不会流动，得到此时每个网格的水平面，地图外的水平面为 0。对于所有没有积水的网格，其海拔值变为相邻的四个网格中水平面的最小值。同时，要求在单次操作中，每个网格海拔的下降最多不能超过 M，超过部分作废。循环执行该操作，求多少次操作后所有网格的海拔变为 0。

数据范围

- 共 20 个测试点，每个点 5 分；
- 测试点 $1 \sim 4$：$1 \leq T \leq 10$，$1 \leq H, W \leq 10$，$1 \leq M \leq 100$，$0 \leq$ 所有海拔 ≤ 100；
- 测试点 $5 \sim 8$：$1 \leq T \leq 50$，$1 \leq H, W \leq 20$，$1 \leq M \leq 100$，$0 \leq$ 所有海拔 ≤ 100；
- 测试点 $9 \sim 20$：$1 \leq T \leq 10$，$1 \leq H, W \leq 20$，$1 \leq M \leq 10^{15}$，$0 \leq$ 所有海拔 $\leq 10^{15}$。

算法分析

首先考虑水平面的问题，显然每个网格当前的水平面等于从地图外到达该网格的所有路径上所有海拔最大值的最小值。因此，对于每次操作，只需进行一次类似最短路的算法即可求得当前的水平面。

下面考虑所需进行的操作次数。对于所有没有积水的网格，若当次操作海拔的可下降值都不小于 M 或者与某个水平面为 0 的网格相邻，则称该网格为自由的。对于某次操作时的情形，若当前所有没有积水的网格都是自由的，则显然将会连续进行若干次类似的操作，直到某个原本积水的网格没有积水为止，而这些操作可以同时进行。

最后考虑该做法的代价。易知，每执行一次上述的连续操作，都会使得不积水的网格数量增加，且不积水的网格永远都不会再积水，因此该类操作最多只会执行 $H \times W$ 次。对于原始的单步操作，通过观察可以发现，每次这种操作都会使得更多的网格变为永远自由的网格，因此该类操作也只会执行至多 $H \times W$ 次。整个算法的时间复杂度为 $\Theta(TH^2W^2 \log(HW))$。

时间复杂度

$\Theta(TH^2W^2 \log(HW))$

空间复杂度

$\Theta(HW)$

2.10 WF2011C-Program within a Program (D8780,P5541)

题目大意

给定一条东西方向的直线，初始位置在原点。题目要求用给定的方式预先编写一段代码，使得按照这段代码执行后，你可以恰好到达某一非负整点 N 处，并结束任务。题中所要求编写的代码由若干条语句组成，每条语句表明了当你正处于某一种特定的状态，并且目前所处整点处的标记为某一特定的值时，你所应该执行的操作。其中操作分为两种：（1）你应该切换到另一种特定状态，并且将当前所处整点处的标记修改为另一种特定值，同时你将会沿着特定的方向（东或西）走一步；（2）你应该立即结束任务。

题目在输入中给定 N，同时假定你的初始状态为 0，并且所有整点处初始的标记值均为 0。要求输出一段由不超过 30 个语句组成的合法代码（即在任意时刻不会有多种选择，也不会没有选择），以完成这个任务。同时，题目还限制了你在任务中执行语句的总次数不能超过 1.5×10^5。

数据范围

- 对于 40% 的数据：$0 \leq N \leq 500$。
- 对于 100% 的数据：$0 \leq N \leq 5000$。

算法分析

Step One

根据题中的叙述，我们首先可以发现：在执行该任务的过程中，除了自身的状态以及整点处的标记，我们不能使用任何的记忆。因此，所有的信息都必须被随时携带。同时由于对语句数极其严格的限制，我们必须充分利用整点处的标记，而不能仅凭自身的状态。另外，操作步数的限制则使我们不能频繁地往返到起点，记录的信息必须一起移动。不过，由于题目中已经给定了 N，所以这很显然是在提醒我们：编写的程序与 N 有关，而不是普遍性的。于是，在编写的程序中我们也要根据 N 的特性来完成。

在上述的简单分析过后，我们首先来解决第一个问题：在任务执行的过程中，我们能够得到的仅仅是当前位置上的标记值，却无从知晓当前所处位置的具体坐标，这也就使得我们无法决定什么时候选择结束任务。因此，我们必须依靠记录的信息来获知距离终点还有多远。于是，我们尝试将 N 的二进制形式从左向右依次标记在各个整点上，然后在每步操作时，我们将标记的这个数减去一，同时将它整体向右移动一位。这样当标记的数减到 0 时，我们就可以知道应该选择结束任务了。

Step Two

接下来，我们来研究上述方法的具体实现。另外，题目中要求状态与标记都要用整数表示，但在之后的讨论中将暂且使用一些符号作为代替，以方便理解（初始状态以及初始标记使用 $\emptyset$ 来表示）。同时，我们暂且使用 $(state, mark)$ 的二元组来表示一种情形，并且使用 $(state, mark, direction)$ 的三元组或者 $Finish$ 来表示一种操作。其中，$state$ 表示自身的状态，$mark$ 表示整点上的标记，$direction$ 表示下一步的方向，$Finish$ 表示结束任务。

首先，第一步要求将 N 的二进制形式标记下来：这一步比较简单，我们可以令自己的状态表示当前已经写到 N 的二进制形式从左往右的第几位，则我们需要 L 条语句来实现：$(pos_i, \emptyset) \to (pos_{i+1}, bit_{a_i}, East), i = 1, 2, \ldots, L-1$，以及 $(pos_L, \emptyset) \to (No, bit_{a_L}, West)$。其中，$L = \lfloor \log N \rfloor + 1$ 表示 N 的二进制形式位数，a_i 表示 N 的二进制形式从左向右第 i 位的数，bit_0 和 bit_1 则分别为表示当前这位是 0 或 1 的两种标记，最后一个语句则是用来进入减一操作（No 状态见下文）。另外，由于起点是原点，我们再用一条语句来进行调整：$(\emptyset, \emptyset) \to (pos_1, bit_0, East)$。这样做也可以固定二进制表示的位数，并方便后续的处理。

接下来，我们来解决减一以及整体右移一位的操作：根据这两个操作的性质，我们选择每次从右向左遍历整个二进制表示，以完成减一的操作，然后再从左往右来完成整体右移一位的操作。因此，

- 对于减一操作，我们共需要 4 个语句：$(owe, bit) \to (owe', bit', West), owe(') = Yes/No, bit(') = bit_0/bit_1$，其中 Yes 表示减一还未完成，即需要向前一位借一，No 则表示减一已经完成。
- 对于右移操作，我们共需要 4 个语句：$(last, now) \to (now, last, East), last(now) = bit_0/bit_1$，表示将前一位的数作为当前位置的标记，将当前位置的数作为状态传递下去。
- 在减一操作中，我们还需要 2 个语句来处理边界情况：$(No, \emptyset) \to (\emptyset, \emptyset, East)$ 和 $(Yes, \emptyset) \to Finish$。其中，第一个语句表示一次完整的减一操作已经完成，转入右移操作；第二个语句则说明任务应当立即结束，因为这种情形只有当所写下的数被减到 0 才会发生。
- 对应的，在右移操作中我们也还需要 3 个语句来衔接：$(\emptyset, bit_0) \to (bit_0, \emptyset, East)$、$(bit_0, \emptyset) \to (Yes, bit_1, West)$ 和 $(bit_1, \emptyset) \to (No, bit_0, West)$。其中第一个语句用来处理二进制表示最高位的右移，后两个语句则用来转入减一操作。

通过上述的方法，当 $N = 5000$ 时，我们只需要 27 个语句就可以完成任务，并且总的操作次数约为 $15 \times 2 \times 5000 = 150000$。这一解法已经可以通过本题，时间复杂度为 $\Theta(\log N)$。

Step Three

其实在本题的限制之内，操作的总步数还可以变得更优：

- 在之前的算法中，我们将 N 改用 k 进制表示，则 $L = \lfloor \log_k N \rfloor + 1$，且初始化同样需要 $L + 1$ 个语句（仍然在最高位前加一个 0）。
- 在减一的操作中，我们总共需要 $2k + 2$ 个语句；而在右移操作中，我们便需要总共 $k^2 + k + 1$ 个语句。
- 因此，当我们选取 k 进制时，总共需要的语句个数为 $L + 1 + 2k + 2 + k^2 + k + 1 = \lfloor \log_k N \rfloor + k^2 + 3k + 5$，而总的操作次数则约为 $2N(\log_k N + 3)$。

由上面的分析发现，我们还可以选取 $k = 3$，此时正好需要 30 个语句，并且总共只有大约 100000 步操作。当选取 k 进制时，整个算法的时间复杂度为 $\Theta(\log_k N + k^2)$。

时间复杂度

$\Theta(\log_k N + k^2)$

空间复杂度

$\Theta(1)$

2.11 WF2013D-Can't stop (D9206,P5792)

题目大意

给定 N 个正整数集合，每个正整数集合包括 D 个正整数。要求选取不超过 k 个正整数，使得某一段连续区间内的所有正整数集合均包含其中至少一个正整数。求最大可能的区间长度。

数据范围

- $1 \leq T \leq 10$；
- $1 \leq D \leq 4$；
- $1 \leq$ 每个$die\ roll \leq 10^5$；
- 对于最多 6 个 $case$，$1 \leq N \leq 10^5$；
- 对于其他所有的 $case$，$1 \leq N \leq 10^3$；
- $k \leq 3$。

算法分析

由于本题的 D 和 k 都很小，因此不妨直接在暴力的基础上进行改进。考虑所选区间的左端点为 i，初始时选取的正整数集合为空，接下来执行如下算法：

- 假设当前共选取了 x 个正整数，则根据贪心我们显然应该首先利用已有的正整数尽可能地向右延伸区间；
- 若当前区间延伸到第 j 个正整数集合时失败，则选取的下一个正整数必然属于第 j 个正整数集合；
- 此时，若新选取的正整数 p 也属于第 $i-1$ 个正整数集合，则易知该种状态在此前必然已经被枚举过，因此不再需要重复执行；
- 显然可以证明，能够使得区间向右延伸到第 i 个正整数集合的左端点个数只有 $\Theta(D^k)$ 个，因此该算法的总代价可以承受。

通过上述分析可知，我们只需要在暴力的基础上加上适当的剪枝即可解决本题。整个算法的时间复杂度为 $\Theta(TD^kn)$。

时间复杂度

$\Theta(TD^kn)$

空间复杂度

$\Theta(Dn)$

2.12 WF2014E-Allergy Testing (D8760,P5591)

题目大意

给定 n 种食物，已知其中有且仅有一种食物会过敏。每次实验可以吃其中的任意多种食物，A 天后可以得知实验结果，若发生过敏则总共需要等待 B 天。求最少需要多少天可以找出过敏的食物。

数据范围

- 50% 的数据满足：$1 \leq A \leq B \leq 100$；
- 100% 的数据满足：$1 \leq n \leq 10^{15}$，$1 \leq A \leq B \leq 10^{12}$，$1 \leq T \leq 10$。

算法分析

首先可以发现，若考虑实验进行的天数则可以方便地通过递推求得 x 天可以鉴别的食物种数的最大值，即：

$$dp[x] = \begin{cases} 1 & x < A \\ dp[x-A] + dp[x-B] & x \geq A \end{cases}$$

通过简单的分析可知，上述的递推构成了一个类似于二叉树的结构，且只有高度差为 A 或 B 的两类边。在这样的结构中，易知叶子节点的个数就是答案。同时，每个叶子节点都可以被 A 和 B 构成的序列唯一表示。因此，我们只需统计有多少个满足条件的 AB 序列即可。

易证，任意一个总高度在 $x-B$ 和 x 之间的 AB 序列均满足条件，且没有遗漏。因此，由于 B 的个数只可能达到 $\Theta(\log n)$ 级别，所以可以直接用组合数计算即可。另外，由于在二分时可能结果过大，此时只需用 *long double* 得出大致的数量级即可。整个算法时间的复杂度为 $\Theta(\log^3 n)$。

时间复杂度

$\Theta(T \cdot \log^3 n)$

空间复杂度

$\Theta(1)$

Chapter 3

USACO

3.1 2005Dec1-Cow Patterns (D9187,P5942)

题目大意

给定长度为 N 的母串以及长度为 K 的模式串，两个串中的所有元素均为 $1 \sim S$ 以内的正整数。定义两个串匹配当且仅当两个串中对应元素的相对大小关系完全一致。求所有匹配位置。

数据范围

- 100% 的数据满足：$1 \leq N \leq 100,000$，$1 \leq K \leq 25,000$，$1 \leq S \leq 25$。

算法分析

显然易证，对于本题而言，KMP 的相关性质仍然成立，只不过我们需要修改两个元素匹配的判定函数如下：

- 假设两个串中的带匹配元素分别为 c_1 和 c_2；
- 两个串中小于 c_1 的元素个数与小于 c_2 的元素个数必须对应相等；
- 两个串中恰好等于 c_1 的元素个数与恰好等于 c_2 的元素个数同样必须对应相等。

对于这一问题，只需在执行 KMP 的过程中同时维护权值树状数组即可解决。整个算法的时间复杂度为 $\Theta((N+K)\log S)$。

时间复杂度

$\Theta((N+K)\log S)$

空间复杂度

$\Theta(N+K)$

3.2 2007Dec3-Best Cow Line (D9167,P6042)

题目大意

给定一个长度为 n 的仅由大写字母组成的字符串，每次操作可以取出最左端或者最右端的一个字符。求通过这种重排方式可能得到的字典序最小的方案。

数据范围

- $n \leq 30000$。

算法分析

考虑当前剩余的字符串为区间 $[l, r]$，下面分两种情况进行讨论：

- 若 $st_l \neq st_r$，则显然选择字典序较小的一端；
- 若 $st_l = st_r$，则通过归纳证明，我们需要寻找最小的 k，使得 $st_{l+k} \neq st_{r-k}$；若 $st_{l+k} < st_{r-k}$，则必然先取 st_l，反之则先取 st_r。

通过上述分析可知，我们只需比较以 l 为起点的后缀和以 r 为起点的前缀的字典序即可。对于这一问题，只需要借助二分以及哈希即可解决。事实上，由于难以构造出特别强的数据，在比较字典序的时候暴力匹配的代价也可以承受，只需稍微优化一下常数即可。整个算法的时间复杂度为 $\Theta(n \log n)$。

时间复杂度

$\Theta(n \log n)$

空间复杂度

$\Theta(n)$

3.3 2008Mar1-Land Acquisition (D8801,P5497)

题目大意

给定 N 块矩形的土地，每块土地的长和宽分别为 L_i 和 W_i。要求进行若干次购买操作，以买下所有的土地。对于每次购买操作，可以选择同时购买若干块土地 $a_1, a_2, \ldots, a_k$，则该次购买的代价为 $max\{L_{a_i}\} \times max\{W_{a_i}\}$。要求最小化购买所有土地的总代价。

数据范围

- 对于 10% 的数据，$N \leq 10$；
- 对于 40% 的数据，$N \leq 2000$，且不存在 i 和 j，使得 $W_i \leq W_j$ 且 $L_i \leq L_j$；
- 对于 60% 的数据，$N \leq 2000$；
- 对于 100% 的数据，$N \leq 50000$。

算法分析

对于这个问题，我们首先可以发现：如果存在 i 和 j，使得 $W_i \leq W_j$ 且 $L_i \leq L_j$，则第 i 块土地总是可以和第 j 块土地一起购买，而无需付出任何额外的代价。因此，我们可以在预处理中通过一次排序筛选出真正可能对答案有影响的土地，即这些土地的宽度应该随着长度的递减而递增。基于这一化简，在接下来的讨论中我们假定：对于任意 $i < j$，必然满足 $L_i > L_j$ 以及 $W_i < W_j$。

显然，在排序过后，我们每次购买的土地必定是其中连续的一段。因此，我们考虑记 $dp[i]$ 为购买前 i 块土地所需要的最小总代价，则我们有如下的 DP 方程：

$$dp[i] = min\{dp[j] + L_{j+1} \times W_i | 0 \leq j < i\}$$

对于这个 DP 方程，我们可以考虑使用斜率优化的方法来解决。对于位置 i 的两个不同的决策点 j 和 k，若 $j < k$ 且在 j 处决策不优于在 k 处决策，则必然有：

$$\begin{aligned} & dp[j] + L_{j+1} \times W_i \geq dp[k] + L_{k+1} \times W_i \\ \Leftrightarrow \quad & dp[k] - dp[j] \leq W_i \times (L_{j+1} - L_{k+1}) \\ \Leftrightarrow \quad & \frac{dp[k]-dp[j]}{L_{j+1}-L_{k+1}} \leq W_i \\ \Leftrightarrow \quad & \frac{dp[j]-dp[k]}{L_{j+1}-L_{k+1}} \geq -W_i \end{aligned}$$

根据上述的推导，由于 W_i 单调递增，所以如果在当前位置，决策点 k 已经优于决策点 j，则在之后的所有位置上，决策点 k 恒优于决策点 j。因此，我们只需要将 $(L_{i+1}, dp[i])$ 视作平面上的点，然后用单调队列维护一个凸壳即可解决本题。整个算法的时间复杂度为 $\Theta(N)$。

时间复杂度

$\Theta(N)$

空间复杂度

$\Theta(N)$

3.4 2008Nov4-Toys (D8749,P5630)

题目大意

已知对于连续 D 天，每天对玩具的需求量为 T_i，要求每天所使用的玩具必须是新购进的或者是经过消毒的。给定初始时买入玩具的单位价格为 T_c，同时提供两种消毒服务，第一种方式需要经过 N_1 天才能完成，单位价格为 C_1，第二种方式需要经过 N_2 天才能完成，单位价格为 C_2。求满足每天需求的最小总代价。

数据范围

- $1 \leq D \leq 100000$，$1 \leq T_i \leq 50$，$1 \leq T_c \leq 60$；
- $1 \leq N_1 \leq D$，$1 \leq N_2 \leq D$，$1 \leq C_1 \leq 60$，$1 \leq C_2 \leq 60$。

算法分析

首先不难发现，所有的买入都可以在最开始时进行。假设总共购进了 X 个玩具，设利用 X 个玩具满足所有需求的最小总代价为 $G(X)$。同时，设当购进的玩具数为 X 时，包括买入费用在内的最小总代价为 $F(X)$（若利用 X 个玩具无论如何都无法满足所有需求，则令 $F(X) = G(X) = \infty$）。易知，$F(X) = G(X) + T_c \times X$。下面证明，$F(X)$ 为单峰函数。

- 要证明 $F(X)$ 为单峰函数，只需证明 $F'(X)$ 具有单调性。由于 $F'(X) = G'(X) + T_c$，而 T_c 为一个常数，因此只需证明 $G'(X)$ 的单调性即可；
- 根据之前的定义，$G'(X)$ 表示，当购进的玩具总数从 X 增加一个到 $X+1$ 时，所有消毒操作最小总代价的减少量。因此，我们需要证明，玩具数从 X 变为 $X+1$ 时比玩具数从 $X+1$ 变为 $X+2$ 时，消毒总代价的减少量更大。这一结论较为显然，因为如果后加入的玩具减少的代价比先加入的玩具减少的代价更多，则必然可以令先加入的玩具代替后加入的玩具，因此每加入一个玩具所减少的代价不会更多。

在得到 $F(X)$ 的单峰性之后，我们便可以通过三分的方式来寻找最优解，因此问题转化为求解当玩具数一定时，消毒操作的最小总代价。对于这一问题，我们可以借助贪心的思想来解决。

- 假设 $N_1 \geq N_2$，即第一种方式为慢洗，第二种方式为快洗，则 $C_1 \leq C_2$，否则慢洗就没有意义了；
- 对于第 i 天，如果当前剩余的新玩具数量足够，则显然全都使用新玩具更优；
- 如果当前剩余的新玩具数量不足，则对于空缺的部分，就需要消毒曾经使用过的玩具来弥补；
- 不难发现，可以进行慢洗的玩具处于连续的若干天中，可以进行快洗的玩具同样处于连续的若干天中；
- 对于可以进行慢洗的玩具，它具体在哪天被使用已经没有意义，只需关心可以进行慢洗的玩具的总数。如果可以进行慢洗的玩具总数足够，则所有的空缺都应该由慢洗来弥补；

- 如果可以进行慢洗的玩具总数依然不足以弥补空缺，则必须对一部分玩具进行快洗。对于连续若干天中可以进行快洗的玩具，由于先使用的玩具可以更早地被慢洗，所以其价值更大，因此应该尽量将后使用的玩具进行快洗；
- 整个过程只需借助单调队列即可实现。

通过上述分析，只需要三分答案并进行贪心即可解决本题。整个算法的时间复杂度为$\Theta(D\log(\sum T_i))$。

时间复杂度

$\Theta(D\log(\sum T_i))$

空间复杂度

$\Theta(D)$

3.5 2008Open3-Cow Neighborhoods (D8742,P5665)

题目大意

给定二维平面上的 N 个点，任意两个点之间存在一条边，当且仅当这两个点之间的曼哈顿距离不超过 C。求图中的连通块个数以及最大的连通块的大小。

数据范围

- $1 \leq N \leq 10^5$，$1 \leq C \leq 10^9$；
- $1 \leq x, y \leq 10^9$。

算法分析

首先，由于曼哈顿距离与点坐标的两维都相关，较为难以处理，因此我们尝试先对点进行预处理。考虑将点 (x, y) 变换为点 $(x+y, x-y)$，则易证任意两点 (x_1, y_1) 与 (x_2, y_2) 之间的曼哈顿距离 $d = |x_1 - x_2| + |y_1 - y_2|$ 等价于变化后两点 (x'_1, y'_1) 和 (x'_2, y'_2) 之间的切比雪夫距离 $d' = max\{|x'_1 - x'_2|, |y'_1 - y'_2|\}$。具体证明如下：

$$\begin{aligned} d &= |x_1 - x_2| + |y_1 - y_2| \\ &= max\{x_1 - x_2 + y_1 - y_2, x_1 - x_2 + y_2 - y_1, x_2 - x_1 + y_1 - y_2, x_2 - x_1 + y_2 - y_1\} \\ &= max\{x'_1 - x'_2, y'_1 - y'_2, y'_2 - y'_1, x'_2 - x'_1\} \\ &= max\{|x'_1 - x'_2|, |y'_1 - y'_2|\} \\ &= d' \end{aligned}$$

通过上述转化，则两点之间有边的条件变为两点间的切比雪夫距离不超过 C，即两点横纵坐标之差中的较大值不超过 C。对于这一问题，考虑借助分治法来解决。若将当前处理的所有点按照其横坐标等分后递归进行，则分治中的每一层均只需处理跨过中间分界线的连边情况。

因此，接下来我们主要考虑分治中合并的过程。又由于对称性，我们只需考虑位于中线右侧的每个点与左侧点集的连边情况。

- 首先，易知所有距离中线超过 C 的点均不可能与另一部分中的点连边；
- 其次，对于右侧的某一点 p，则左侧可能与 p 存在连边的点 q，必然满足 p 与 q 的纵坐标之差不超过 C。因此对于点 p 而言，可能的点 q 必然处于一段连续的纵坐标区间内，所以若我们将所有待处理的点按照纵坐标排序，则可能的点 q 必然是连续的若干个点；
- 由上述粗略的分析可知，与点 p 可能有连边的点在平面坐标系中表现为一块 $C \times 2C$ 的矩形。对于这一矩形区域，考虑分为上下两部分来处理，即按照点 p 的纵坐标再进行一次分割；
- 由于分治的性质，易知被分成的两部分中，同一区域内的点必然已经属于同一个连通块内，因为这些点的横纵坐标之差均不超过 C。换言之，这一结论表明，对于同一区域内的所有点，尽管与点 p 的连边可能存在若干条，但真正有意义的最多只有一条，即只需判断该区域中的最右点是否与点 p 有边相连；

- 对于这一分治的过程，统计答案只需借助两个单调队列即可实现，而按点的纵坐标排序则只需在分治的同时进行归并排序即可。

借助上述分析，我们只需在分治时借助并查集来维护连通性即可解决本题，整个算法的时间复杂度为 $\Theta(N \log N)$。

时间复杂度

$\Theta(N \log N)$

空间复杂度

$\Theta(N)$

3.6 2009Mar2-Cleaning Up (D9230,P5622)

题目大意

给定一个长度为 n 的正整数序列，每个正整数均属于 $1 \sim m$。要求将整个序列分割成若干段，每一段的权值为该段内正整数种类数的平方。求权值总和的最小值。

数据范围

- 对于 30% 的数据，$1 \leq n \leq 5000$；
- 对于 100% 的数据，$1 \leq m \leq n \leq 40000$。

算法分析

首先可以发现，即使直接将每个数都单独分为一组，则权值和也只有 n，因此最优解中每一段内正整数的种类数必然不超过 $\sqrt{n}$。因此，若设计 DP 状态为 $dp[i]$，表示对前 i 个正整数进行分割的最小代价，则有效的转移点显然只有 $\sqrt{n}$ 个，并且这些点只需借助队列维护一下即可。整个算法的时间复杂度为 $\Theta(n\sqrt{n})$。

时间复杂度

$\Theta(n\sqrt{n})$

空间复杂度

$\Theta(n)$

3.7 2009Open3-Tower of Hay (D8779,P5542)

题目大意

给定 N 件只有宽度 w_i 不同的物品，要求按照顺序从下往上依次按层放置，且一旦开始放置新的一层，则不允许再放置在下面的层中。同时，题目要求保证每一层的宽度都必须不小于上一层的宽度，求用所有的物品最多可以放置多少层。

数据范围

- $1 \leq N \leq 100000$。
- $1 \leq w_i \leq 10000$。

算法分析

首先，为了方便讨论以及解题，我们不妨将整个序列倒置，则题中的要求转化为每一层的宽度都必须要不小于下一层的宽度。其次，我们需要通过下面这个结论的帮助来转化题目：若前 i 件物品存在一种放置层数最多的方案，则该种方案也一定是所有合法的方案中，最高层最窄的方案。

对于这个结论，我们可以有如下的归纳证明：首先，若我们记 $dp[i]$ 为前 i 件物品所能放置的最多层数，则对于任意 $i < j$，$dp[i] \leq dp[j]$ 显然必定成立，因为我们可以将多余的物品全部放置在最高层。有了这个最基本的结论，则对于任意 $j < k$，若 j 和 k 都是位置 i 的一个合法的决策点（j 是位置 i 的一个合法的决策点，当且仅当存在一种合法的放置方案，使得该方案的最高层放置的是第 $j+1$ 到 i 件物品），那么 $dp[j] \leq dp[k]$ 恒成立。同时，由于归纳假设，前 $j(k)$ 件物品的最优方案一定是最高层最窄的方案，所以在该方案的基础上再放置第 $j+1(k+1)$ 到 i 件物品作为新的一层一定是合法的，因此一定有 $dp[j]+1 \leq dp[k]+1$，即决策点 k 恒优于决策点 j。

通过上述的讨论，我们便只需对于每个位置找到其下标最大的可行决策点。而对于这个问题，若我们记 $f[i]$ 为前 i 件物品的最优方案中最高层的宽度，记 $sum[i]$ 为前 i 件物品宽度之和，则对于一个在位置 i 时可行的决策点 j 必有 $sum[i]-sum[j] \geq f[j]$，即 $sum[i] \geq sum[j]+f[j]$。由于 $sum[i]$ 单调递增，因此我们可以对 $sum[i]+f[i]$ 的值用单调队列来维护，然后每次选取下标最大的可行决策点即可。整个算法的时间复杂度为 $\Theta(N)$。

时间复杂度

$\Theta(N)$

空间复杂度

$\Theta(N)$

3.8 2010Mar3-StarCowraft (D9218,P5736)

题目大意

给定 n 组形如 $ax + by + cz < 0$ 或 $ax + by + cz > 0$ 的限制条件，同时要求 x、y 和 z 均为正实数，并且两两之间的比值不超过 100。给出 m 组询问，对于每组询问 a_i、b_i 和 c_i，分别询问是否存在满足所有限制条件的 x、y 和 z，同时使得 $a_ix + b_iy + c_iz < 0$ 或者 $a_ix + b_iy + c_iz = 0$ 或者 $a_ix + b_iy + c_iz > 0$。

数据范围

- 对于 50% 的数据，$n \leq 100$；
- 对于 100% 的数据，$n \leq 300$，$m \leq 2000$。

算法分析

本题单从表面上看是一道有关半空间交的问题，但通过观察不难发现，由于所有限制条件以及询问中均没有常数项，所以我们可以直接通过转化为相互之间的比值来降维。因此，本题可以被转化为半平面交问题来解决，且由于限制条件和询问的数量以及坐标范围较小，所以只需暴力构造并判交即可。整个算法的时间复杂度为 $\Theta(n^3 + nm)$。

时间复杂度

$\Theta(n^3 + nm)$

空间复杂度

$\Theta(n^2)$

3.9 2010Dec3-Threatening Letter (D8788,P5520)

题目大意

给定两个长度分别为 N 和 M 的字符串 S 和 T，要求将 S 分割成若干个子串，使得每一段子串均是 T 的子串。求最少需要分成多少段。

数据范围

- $N, M \leq S$；
- 对于 30% 的数据，$S = 500$；
- 对于 100% 的数据，$S = 50000$；
- 数据的生成有随机因素。

算法分析

首先不难证明，本题直接进行贪心的正确性显然，即每次操作我们直接尽可能地找到 S 中最长的一段前缀，使得该段前缀是 T 子串。对于转化后的问题，则只需要借助后缀数组来求解两个字符串的最长公共前缀即可。整个算法的时间复杂度为 $\Theta((N+M)\log(N+M))$。

时间复杂度

$\Theta((N+M)\log(N+M))$

空间复杂度

$\Theta(N+M)$

3.10 2010Open3-Triangle Counting (D9211,P5765)

题目大意

给定平面直角坐标系中的 n 个已知点，求由这些点构成的所有三角形中，有多少个三角形包含坐标原点。

数据范围

- $1 \leq n \leq 10^5$;
- $-10^5 \leq X_i, Y_i \leq 10^5$;
- 不会有两点的直线经过原点，特别地，不会有点恰好是原点。

算法分析

通过观察可以发现，不包含原点的三角形个数比较容易统计。易知，对于任意一个不包含原点的三角形而言，必然满足三个顶点的极角序处于某个不超过 π 的区间内。因此，我们可以先将所有点按照极角序排序，则以该点的极角序为一个端点的极角区间所包含的点必然是连续的一段，同时由于单调性显然，因此只需扫描一遍即可。整个算法的时间复杂度为 $\Theta(n)$。

时间复杂度

$\Theta(n)$

空间复杂度

$\Theta(n)$

3.11 2012Mar3-Cows in a Skyscrape (D9185,P5953)

题目大意

给定 n 头奶牛，第 i 头奶牛体重为 C_i，同时已知电梯的最大承重量为 m。求最少需要多少次才能将所有奶牛全部运完。

数据范围

- 对于 10% 的数据，$n \leq 5$；
- 对于 100% 的数据，$n \leq 18$。

算法分析

对于本题，显然应该考虑进行 DP。不妨设 DP 状态为 $dp[i][S]$，表示对于奶牛集合 S 共运输 i 次，最多能运输多少重量，设 $tot[S]$ 为奶牛集合 S 的总重量，则转移有以下两种情形：

- 对于任意一个奶牛集合 S，若满足 $tot[S]-dp[i][S] \leq m$，则只需要再进行一次运输即可，即有 $dp[i+1][S] = tot[S]$；
- 对于任意一个状态 $dp[i][S]$ 而言，显然还可以通过增加任意一个头新奶牛来转移。

易知，通过以上两种转移，则所有可能的最优方案均被覆盖过。整个算法的时间复杂度为 $\Theta(2^n n^2)$。

时间复杂度

$\Theta(2^n n^2)$

空间复杂度

$\Theta(2^n n)$

3.12 2012Dec1-Gangs of Istanbull/Cowstantinople (D8754,P5623)

题目大意

已知有 M 个帮派，第 i 个帮派共有 a_i 头奶牛，总共有 N 头奶牛。要求 N 头奶牛依次进入牧场，若此时牧场中为空或者只有同一帮派的奶牛，则该头奶牛将会停留在牧场中。否则，当前控制牧场的帮派将有一头奶牛与该头奶牛一起离开牧场。求帮派 1 是否能够最终控制牧场，若能，则最后最多剩下几头帮派 1 的奶牛。同时，要求在满足前两问最优的前提下给出一组字典序最小的奶牛序列。

数据范围

- $1 \leq N \leq 1000000$，$1 \leq M \leq N$。

算法分析

首先易知，若要使得帮派 1 剩余的奶牛最多，则应该使得其余所有帮派的奶牛尽可能地结对离开牧场。不难证明，若其余帮派中存在某个帮派的奶牛总数超过半数，则应该令剩下的所有奶牛均与该帮派的奶牛结对；否则，若剩余奶牛的总数为奇数，则可以做到只剩下一头奶牛，若为偶数，则可以做到全部结对。因此，我们可以直接贪心地构造出最终的奶牛序列。整个算法的时间复杂度为 $\Theta(N)$。

时间复杂度

$\Theta(N)$

空间复杂度

$\Theta(N)$

3.13 2012Dec2-First! (D8762,P5588)

题目大意

给定 n 个仅由小写字母构成的字符串，判断通过重新安排字母表的优先级，每个字符串是否可能成为字典序最小的字符串。

数据范围

- 对于 20% 的数据，$1 \leq n \leq 10$，$1 \leq$ 字符串总长度 ≤ 200。
- 对于 50% 的数据，$1 \leq$ 字符串总长度 ≤ 100000。
- 对于 100% 的数据，$1 \leq n \leq 30000$，$1 \leq$ 字符串总长度 ≤ 300000。
- 字符串保证不重复。

算法分析

对于任意两个字符串 str_i 和 str_j，若记 $lcp(i,j)$ 为这两个字符串的最长公共前缀，则对于任意一个字符串 str_i 而言，其能够成为字典序最小的字符串的充要条件是：

- 不存在任何一个其它字符串 str_j，使得 str_j 是 str_i 的前缀。
- 对于任何一个其它字符串 str_j，都必须要满足字符 $str_i[lcp(i,j)+1]$ 的优先级要高于字符 $str_j[lcp(i,j)+1]$ 的优先级。

因此，对于任意一个字符串 str_i 而言，若上述条件中所有字符之间优先级的大小关系可以被同时满足，则该字符串可能成为字典序最小的字符串，否则就不可能。对于这个问题，我们可以将所有字符串建成字典树，每次在扫描一个字符串时，将该字符串的每一位字符所必须满足的优先级关系记录下来，最后通过判断优先级关系中是否存在环来检验合法性。整个算法的时间复杂度为 $\Theta(n|S|^2)$，其中 S 表示字符集。

时间复杂度

$\Theta(n|S|^2)$

空间复杂度

$\Theta(|S|\sum length_i)$

3.14 2013Open1-Photo (D9165,P6044)

题目大意

给定 N 头奶牛，已知 M 组限制，第 i 组限制 $[a_i, b_i]$ 表示第 a_i 头到第 b_i 头奶牛中恰好有一头奶牛带有斑点。求至多有多少头带斑点的奶牛。

数据范围

- 对于 20% 的数据，$N \leq 2 \times 10^2$，$M \leq 10^2$；
- 对于 60% 的数据，$N \leq 2 \times 10^3$，$M \leq 10^3$；
- 对于 100% 的数据，$N \leq 2 \times 10^5$，$M \leq 10^5$。

算法分析

显然，该题考虑借助 DP 来解决，不妨设计 DP 状态为 $dp[i]$，表示当第 i 头奶牛必然有斑点时，前 i 头奶牛中最多有多少头奶牛有斑点。接下来考虑转移，易证，能够从 j 转移到 i 的充要条件为：

- 不存在任意一组限制 $[a_k, b_k]$，使得 $j < a_k$ 且 $b_k < i$ 同时成立；
- 不存在任意一组限制 $[a_k, b_k]$，使得 $a_k \leq j$ 且 $i \leq b_k$ 同时成立。

显然，满足上述条件的可转移区间的两个端点均是单调的，因此只需借助单调队列即可降低 DP 的总代价。整个算法的时间复杂度为 $\Theta(N + M)$。

时间复杂度

$\Theta(N + M)$

空间复杂度

$\Theta(N + M)$