2019 CCPC Harbin Site Editorial

Prepared by Zhejiang University

2019 年 10 月 13 日

A Artful Paintings

Shortest judge solution: 2161 Bytes.

设 f_i 表示前 i 个位置有多少个格子被涂黑，令 $x = f_n$，考虑差分约束系统建图：

- $f_{i-1} - f_i \le 0$：$i \to i-1$，边权为 0。
- $f_i - f_{i-1} \le 1$：$i-1 \to i$，边权为 1。
- 对于每条第一类限制有 $f_{l-1} - f_r \le -k$：$r \to l-1$，边权为 $-k$。
- 对于每条第二类限制有 $f_r - f_{l-1} \le -k + x$：$l-1 \to r$，边权为 $-k+x$。
- $f_n - f_0 \le x$：$0 \to n$，边权为 x。
- $f_0 - f_n \le -x$：$n \to 0$，边权为 $-x$。

注意到 x 越大这些约束条件越容易满足，因此二分 x 的值然后 SPFA 判断是否存在负环即可，这样就得到了一个时间复杂度为 $O(nm \log n)$ 的算法。

考虑每一个环，设其边权和为 $ax + b$，那么有 $ax + b \ge 0$，即 $x \ge -\frac{b}{a}$。只要找到 $\frac{b}{a}$ 最小的环即可求出 x 的取值范围。这很类似于 “边权平均数最小环” 问题。回顾求平均数最小环的算法：令 $g_{i,j}$ 为从 0 号点出发经过 i 条边到达 j 的最短路，则边权平均数最小值为：

$$ans = \min_{i=0}^{n} \left\{ \max_{j=0}^{n} \frac{g_{n+1,i} - g_{j,i}}{n+1-j} \right\}$$

对于这个问题也是类似的，去掉边权为 $-x$ 的那条边，令 $g_{i,j}$ 为从 0 号点出发到达 j 的路径中，$a = i$ 时 b 的最小值，则不经过边权为 $-x$ 的那条边的最小环为：

$$ans_1 = \min_{i=0}^{n} \left\{ \max_{j=0}^{n} \frac{g_{n+1,i} - g_{j,i}}{n+1-j} \right\}$$

而经过边权为 $-x$ 的那条边的环一定是从 0 走到 n 的某条路径加上 $-x$ 这条边，因此这种情况的最小环为：

$$ans_2 = \min_{i=2}^{n+1} \frac{g_{i,n}}{i-1}$$

最终 x 的最小可能值为 $-\min(ans_1, ans_2)$，假设求出了所有 $g_{i,j}$，那么通过上述式子计算答案显然是 $O(n^2)$ 的。

考虑如何计算 g，由 g_i 转移到 g_{i+1} 非常容易，只需要枚举边权包含 x 的边进行转移，而同层 g_i 内部边权不包含 x 的边的松弛则比较棘手。方便起见固定 i，设 h_j 表示 $g_{i,j}$。

首先从后往前递推：$h_i = \min(h_i, h_{i+1})$，再从前往后递推：$h_i = \min(h_i, h_{i-1} + 1)$，那么因为剩下的边都是从大点连向小点的，所以 h_n 的值已经确定。按照 n 到 0 的顺序逐一确定每个 h_i 的值。在确定 h_i 的值时，只需要判断 h_i 是 $h_{i+1} - 1$ 还是 h_{i+1}，这里有两种情况：

- $h_i = \min(h_i, h_{i+1})$，可以 $O(1)$ 确定。
- $h_i = \min(h_i, h_j + i - j)$，其中 $j < i$，这是由后面往前松弛的边造成的。注意到 $h_n - h_i$ 在 $[0, n+1]$ 范围内，因此对于每种值使用数组计数即可轻松 $O(1)$ 确定是否存在这样的 j 使得 $h_i = h_{i+1} - 1$。

确定完 h_i 的值后，枚举所有从 i 出发的边，用 $h_i - k$ 更新相应的 h_{l-1} 即可。这样我们就在 $O(n(n+m))$ 的时间内求出了 g。

最后我们得到了一个 $O(n(n+m))$ 的算法，但是因为考虑到难度较大以及 $O(nm \log n)$ 算法中的 SPFA 剪枝空间很大，所以允许 $O(nm \log n)$ 的算法通过。

B Binary Numbers

Shortest judge solution: 1108 Bytes.

$F(a, b)$ 其实就是把 a 和 b 看作二进制串求最长公共前缀 (LCP)。如果固定 i，假设 $j < i$，那么越小的 j 与 i 的 LCP 越短，同理 $j > i$ 时越大的 j 与 i 的 LCP 越短。因此对于每一个区间，只需要满足如下两条限制：

- $\text{LCP}(a_i, l_i) \geq \text{LCP}(a_{i-1}, l_i)$
- $\text{LCP}(a_i, r_i) \geq \text{LCP}(a_{i+1}, r_i)$

动态规划，设 $f[i][j][k]$ 表示考虑了前 i 组，$\text{LCP}(a_i, l_{i+1}) = j$，$\text{LCP}(a_i, r_i) = k$ 时的贡献和，枚举第 i 组选取哪个数作为 a_i，那么暴力枚举 $f[i-1][][]$ 里合法的 j 和 k 进行转移即可。

期望时间复杂度为 $O(2^m m^2)$。

C Competition in Swiss-system

Shortest judge solution: 1928 Bytes.

注意到每个选手最多只会有 $O(m)$ 个对手，所以暴力模拟的时间复杂度为 $O(nm^2)$，可以接受。

需要注意的是分数类的实现以及小心地约分使得分子分母不爆 `long long`。

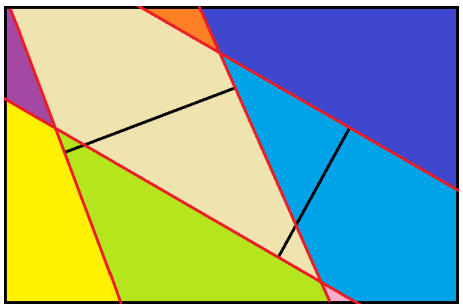
分子不会大于分母，而分母不超过 $lcm(2m+1, 2m+2, \ldots, 3m) \times 3 \times m \leq lcm(33, 34, \ldots, 48) \times 48$，所以可以使用 `long long` 存储。

D Driverless Car

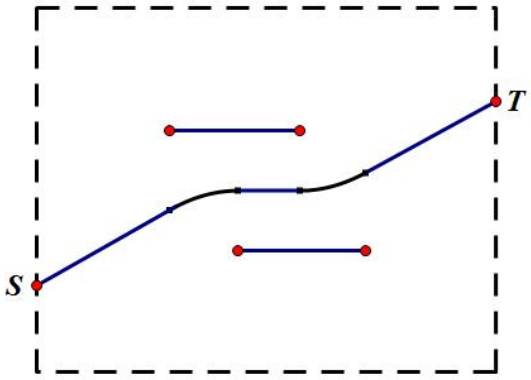
Shortest judge solution: 4727 Bytes.

注意到一定存在合法路线，而且合法路线唯一，所以只需要计算那条路线的总长度。

点到线段的距离可以按照两根垂线把平面分成三个区域，两侧的区域是点到端点的距离，中间的区域是点到直线的距离。找出两条线段的两条垂线，以及矩形的四条边界，那么可以把矩形划分成若干个凸多边形区域，对于每个凸多边形分别求解。



在每个区域中，有三种情况：点到两点距离相等、点到两直线距离相等、点到某点以及到某直线距离相等。



对于第一种情况，只需要计算两点的垂直平分线与该凸多边形的交的长度。

对于第二种情况，只需要计算两直线的角平分线与该凸多边形的交的长度。注意特判直线平行/重合等特殊情况，以及需要注意两条直线有两条角平分线。

对于第三种情况，对应的轨迹是以该点为焦点，直线为准线的抛物线。需要计算抛物线与该凸多边形的交的长度。一个比较简便的方法是先将坐标系旋转使得抛物线开口正向上，然后平移坐标系使得抛物线顶点为 $(0,0)$，最后求出抛物线与凸多边形的每条边的交点的横坐标，按照 x 分段进行曲线积分。

E Exchanging Gifts

Shortest judge solution: 1384 Bytes.

首先 $O(n)$ 递推求出每个序列的长度 len_i，令 $occ(x)$ 表示数字 x 在 s_n 中的出现次数。如果 $2\max\{occ(x)\} > len_n$，则答案为 $2(len_n - \max\{occ(x)\})$，否则答案为 len_n。所以只要找到出现次数最多的数。

对于每个序列 i 维护两个信息：c_i 表示在序列 s_i 里可能的出现次数最多的数是谁，w_i 表示 c_i 的出现次数。那么合并两个序列的时候，如果它们的 c 相等，则将其 w 相加；如果它们的 c 不相等，则保留 w 较大的那个作为 c，并将 w 设置为它们的 w 的差值。

如此在 $O(n)$ 的时间内可以递推求出 s_n 的信息，那么唯一可能的出现次数最多的数只能是 c_n，在 $O(n)$ 时间内递推计算出 c_n 的真实出现次数即可。

F Fixing Banners

Shortest judge solution: 580 Bytes.

签到题，6! 枚举所有情况或者状压 DP 都可以通过。

G Game Store

Shortest judge solution: 846 Bytes.

如果 Bob 不能扔掉石子，那么这是 Nim-K 游戏的 $K = 2$ 的情况，先手必败当且仅当将每一堆石子的石子数看作二进制后，二进制每一位加起来的和都是 3 的倍数，即把每一堆石子的石子数看作 $O(\log a)$ 维的 01 向量后，这些向量在模 3 意义下的和为零向量。

因为每副道具都包含两包相同的石子，所以 Bob 可以选择将每种石子保留 0 份、1 份或者 2 份，这恰好对应模 3 的所有系数。因此 Alice 想要必胜，那么这些向量在模 3 意义下必须线性无关。

问题转化为动态维护权值和最大的线性无关向量组。维护 $O(\log a)$ 个向量，其中第 i 个向量为零向量或权值最大的第 i 位不为 0 的向量。每次加入新的向量时，遍历该向量中非 0 的位，如果那一位对应向量为空，则将该向量放在那个位置，否则将两个向量中权值较小的那一个向量拿出来继续往后消元。

数据范围比较大，因此需要使用位运算来加速模 3 意义下的加法，时间复杂度 $O(\frac{n\log^2 a}{w})$。

H Highway Buses

Shortest judge solution: 3380 Bytes.

因为每个车站的费用都是一次函数，而一次函数的和仍然是一次函数，所以最优解只可能在第一天或者最后一天取到，分别对于这两天计算从 1 出发到达每个站点的最小花费即可。

注意到图连通且 $m \le n + 50$，那么可以将图看作一棵生成树上面多余不超过 51 条非树边。从 1 点开始 Dijkstra，每次从堆中取出 $dis_x + cost_x$ 最小的 x，并用 $dis_x + cost_x$ 去更新所有距离 x 不超过 f_i 条边的点。因为 Dijkstra 取出的 $dis_x + cost_x$ 单调不下降，所以每个点只有第一次被更新的时候才有用，可以考虑更新完后就将其删除。

对于生成树的部分，是经典树分治问题。把点分治的过程记录下来，每个分治结构按到重心的距离从小到大保存一个队列，枚举管辖 x 的 $O(\log n)$ 个分治结构，从对应的队列中不断消

费队首，总计 $O(n\log n)$。

对于非树边的部分，考虑路径经过某条非树边 (u,v) 的情况，那么一定经过了 u 点。令 u 点为关键点，则最多只有 51 个关键点。对于每个关键点 BFS 求出单源最短路，同样用队列保存，那么只需要枚举这不超过 51 个关键点，从对应的队列中不断消费队首即可，总计 $O(51n)$。

总时间复杂度为 $O(n(51+\log n))$。

I Interesting Permutation

Shortest judge solution: 543 Bytes.

首先特判 $h_1\neq 0, h_i\geq n$ 以及 $h_i>h_{i+1}$ 的情况，这些情况都是无解。

然后依次考虑 $h_2,h_3,\ldots,h_n$。如果 $h_i>h_{i-1}$，那么 a_i 既可以是前 i 个数的最大值，也可以是前 i 个数的最小值，将答案乘以 2，同时新增中间 $h_i-h_{i-1}-1$ 个空位没填；如果 $h_i=h_{i-1}$，那么需要消耗一个空位，并将答案乘以空位数。

时间复杂度 $O(n)$。

J Justifying the Conjecture

Shortest judge solution: 235 Bytes.

当 $n\leq 5$ 时无解，否则如果 n 是偶数，则可以分解成 2+ 一个偶数，如果 n 是奇数，则可以分解成 3+ 一个偶数。

K Keeping Rabbits

Shortest judge solution: 308 Bytes.

容易发现不管经过了多少天，概率分布都不会改变，所以令 sum 为所有 w 的和，则 $ans_i=w_i+\frac{w_i\times k}{sum}$。

L LRU Algorithm

Shortest judge solution: 863 Bytes.

令缓存无容量限制 $O(n^2)$ 模拟得出每一步后的 LRU 序列，那么对于每个询问只需要检查是否存在某一步后的 LRU 序列以该询问串作为前缀。

求出每个询问串的 Hash 值，那么每模拟一步后，$O(n)$ 预处理出当前 LRU 序列每个前缀的 Hash 值，然后遍历所有询问判断即可。

时间复杂度 $O(n(n+m))$。

另一种方法是将询问串全部插入 Trie 中，每模拟一步后在 Trie 中找到对应前缀，并将沿途经过的前缀都标记为出现过。