IOI2021 国家集训队第一阶段作业试题准备解题报告

长郡中学 彭思进

2020.10

目录

1 Bipartite Blanket

1.1 试题来源

2016-2017 ACM-ICPC, Central Europe Regional Contest (CERC 16) Problem. B

1.2 简要题意与数据范围

给出一个带点权的二分图，定义二分图的一个匹配**覆盖**一个点集当且仅当这个匹配的端点集合包含这个点集。给出限制 t，你需要求出二分图中有多少个点集满足点集中的点权和不小于 t 且存在一个匹配覆盖这个点集。

设二分图两部的点数为 n, m，则 $1 \le n, m \le 20$。点权为正整数且不超过 10^7，$1 \le t \le 4 \times 10^8$。

时间限制：`1s`；空间限制：`512MB`

1.3 知识考察

二分图、Hall 定理、高维前缀和、two-pointer

1.4 定义与约定

设给出的二分图为 G，其左部点构成的集合为 V_l，右部点构成的集合为 V_r，边集为 E。

对点集 S，设集合 $N_G(S)$ 为图 G 中与 S 任意一点相邻的点构成的集合。不产生歧义时忽略下标 G。

对于图 $G(V, E)$ 和 $V' \subseteq V$，设 $G(V')$ 为图 G 在点集 V' 下的导出子图。

1.5 算法介绍

1.5.1 重要的定理

定理 1.1. *对于任意点集 S，设 $S_l = S \cap V_l, S_r = S \cap V_r$，则存在一个匹配覆盖 S 当且仅当存在一个匹配覆盖 S_l 且存在一个匹配覆盖 S_r。*

证明. 必要性是显然的。下文将会通过构造的方式证明充分性。

考虑任一覆盖 S_l 的匹配 M_l 和覆盖 S_r 的匹配 M_r。由于每个点在图 $G'(V_l \cup V_r, M_l \cup M_r)$ 上的度数不超过 2，故 G' 包含若干条链和环。同时 G' 上度数为 2 的点一定属于 S，度数为 0 的点一定不属于 S。

由于 G' 是二分图，所以任意一个环包含偶数条边。在环上交错选边可以保证选出的是一个匹配且环上的每个点均被覆盖。对于边数为奇数的链，也可以从边界开始交错选边达到同样效果。

对于边数为偶数的链，其两个端点同时属于 V_l 或同时属于 V_r。有引理

引理 1.1. *对于一条 G' 上边数为偶数的链，其两端至少有一个节点不属于 S。*

证明. 设 C_l, C_r 分别表示链上属于 V_l, V_r 的点构成的集合。不失一般性地假设链的两端点属于 V_l，那么有 $|C_l| - |C_r| = 1$。

考虑反证，若两端点均属于 S，又度数为 2 的点一定属于 S，则 $C_l \subseteq S_l$，C_l 中每个点在匹配 M_l 上都对应一条出边。而 $|C_l| > |N_{G'}(C_l) = C_r|$，又由鸽巢原理可以知道 C_l 中一定存在两个点满足其在 M_l 中匹配相同的点。这与 M_l 是匹配矛盾，故假设不成立，至少存在一个端点不属于 S。 □

这样可以删去链上的一个不在 S 中的端点，得到一条边数为奇数的链，然后按照上述方法操作。

最后将每一个连通块中所选择的边取并，可以得到一个匹配 M 覆盖 $S_l \cup S_r$。 □

由上述定理，如果对于 V_l 和 V_r 分别得到了所有被至少一个匹配覆盖的子集，那么可以将它们按照权值排序之后使用 two-pointer 算法统计答案。不考虑排序这一部分复杂度为 $O(2^n)$。

下面涉及的两个算法的目的均为找出所有左部点和右部点的满足条件的子集。

1.5.2 利用二分图匹配求解的算法

考虑使用 dfs 枚举所有可能的集合并维护可以覆盖当前集合的任一合法匹配。加入新点时，如果当前点未被覆盖，则使用匈牙利算法进行增广，若增广失败则表示当前集合不能被任一匹配覆盖。

进行一次增广的复杂度是 $O(n^2)$ 的，所以复杂度为 $O(2^n n^2)$。由于增广常数不大，可能可以通过。

1.5.3 利用 Hall 定理求解的算法

注意到判断 S_l 是否可以被覆盖等价于判断 $G(S_l \cup V_r)$ 是否存在对 S_l 的完美匹配。考虑 Hall 定理：

定理 1.2 (Hall 定理)**.** *二分图 (V_l, V_r, E) 存在对 V_l 的完美匹配当且仅当 $\forall V \subseteq V_l, |N(V)| \geq |V|$。*

证明可参考资料 [2]。

对于集合 S，设 $f_S = [|N(S)| \geq |S|], g_S = \wedge_{i \subseteq S} f_S$，其中 $\wedge$ 是逻辑与运算，那么 S 合法当且仅当 $g_S = 1$。由 f_S 计算 g_S 可以由高维前缀和实现，那么问题是计算 f_S，即计算 $N(S)$。n, m 范围不大考虑状压，将每个点的邻居集合用二进制数存储，计算 $N(S)$ 时将 S 中所有点的邻居集合取并即可。

总复杂度为 $O(2^n n)$，可以轻松通过。

1.6 参考资料

[1] cerc2016 presentation: cerc.hsin.hr/2016/tasks/cerc2016_presentation.pdf

[2] HALL'S MATCHING THEOREM: galton.uchicago.edu/~lalley/Courses/388/Matching.pdf

2 Intellectual Property

2.1 试题来源

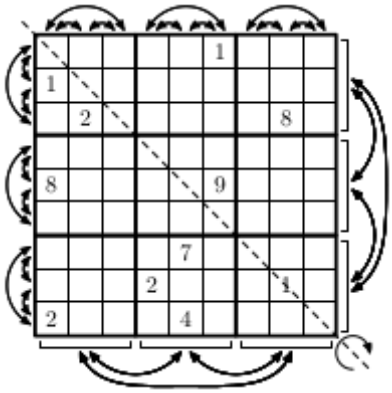
2013-2014 ACM-ICPC, NEERC, Northern Subregional Contest Problem. I

2.2 简要题意与数据范围

给出 n 个 9×9 的数独谜题，每个谜题上部分格子已写上数字，其余格子为空格。你需要对其中任意一对数独判断它们是否**相似**。两个谜题相似当且仅当对第一个谜题以任意顺序施加以下操作若干次可以得到第二个谜题：

- 选择两个数字 x, y，将所有的 x 变为 y，y 变为 x；
- 在 $(1,2,3),(4,5,6),(7,8,9)$ 三个三元组中选择两个交换对应的行；
- 交换同一个三元组里的两行；
- 在 $(1,2,3),(4,5,6),(7,8,9)$ 三个三元组中选择两个交换对应的列；
- 交换同一个三元组里的两列；
- 沿着左上-右下对角线将谜题翻转，此时行列翻转。

如果两个谜题相似，还需要输出一个将第一个谜题变为第二个谜题的操作方案。不要求最小化操作数，但操作数不能超过 10^3。下图描述了所有的单次操作。



$n \leq 20$。

时间限制：`2s`；空间限制：`512MB`

2.3 知识考察

折半枚举、字符串哈希

2.4 定义与约定

为了拓展到更一般的情况，下文中定义 k 表示解决的是 $k^2 \times k^2$ 的问题。在本题中 $k = 3$。

2.5 算法介绍

2.5.1 朴素算法

由于对行和列的操作是相同的，所以若有解则一定存在某个解的翻转操作最多做一次，可以枚举。而其他操作的影响相当于对行下标、列下标和数字施加置换，考虑枚举合法置换进行判断。由于行列操作的特殊性，合法的行列置换只有 $(k!)^{k+1}$ 种，即 k 元组的任意排列以及 k 个 k 元组内部的任意排列。复杂度 $O(n^2(k!)^{2k+2}(k^2)!k^4)$，无法通过。

注意到确定了行列置换后数字的置换是确定的，无需枚举。复杂度降为 $O(n^2(k!)^{2k+2}k^4)$，但仍较大。

2.5.2 对朴素算法的折半枚举优化

可以发现性质：操作是可逆的，即若谜题 P 通过一个操作序列得到了谜题 Q，那么从谜题 Q 倒着做这个操作序列恰好得到谜题 P。这意味着相似是具有传递性的。

同时注意到上述算法在判断 P_i 和 P_j 是否相似时，对 P_i 枚举较多操作复杂度较高，而与 P_j 判断是否相等的复杂度较低，复杂度并不平衡。

考虑使用折半枚举平衡复杂度，即在 P_i 上枚举行置换，在 P_j 上枚举列置换，并将结果存储为两个谜题集合。那么只需要判断这两个谜题集合中是否存在一对谜题在经过数字置换后可以变得一样。

由于相似的传递性，可以考虑对于每个谜题，在其能够通过数字置换得到的谜题中取出一个代表元，两个谜题仅经过数字置换相似当且仅当两个谜题的代表元相同。字典序最小的谜题是一个可行的选择。

具体地，将谜题的 k^2 行首尾相接得到一个长度为 k^4 的字符串，将除了空格外的其他字符按照第一次出现位置从先往后依次替换成 1 至 k^2 中的数字，可以得到通过数字置换能得到的最小字典序的谜题。

接下来只需判断两个集合内是否存在相同谜题，使用字符串哈希对谜题进行处理之后问题变为两个集合是否有交，枚举其中一个集合的元素，对另一个集合建立哈希表在其中进行查询即可。

得到了一对相同的谜题之后构造方案是容易的：只需要把行列元排列通过交换变为两个谜题所枚举的行列排列，并通过数字间的映射关系得到关于数字的操作。不要忘记最初枚举的翻转。总步数不超过 $3k^2-2$，这显然不超过 10^3。

精细地实现复杂度可以做到 $O(n(k!)^{k+1}k^4+n^2(k!)^{k+1})$，可以通过。

3 Gem Island

3.1 试题来源

2018 ACM-ICPC World Finals Problem. D

3.2 简要题意与数据范围

给定一个长度为 n 的序列 a，初始 $a_i = 1$。接下来进行 d 次操作，每次操作有 $\frac{a_i}{\sum_{j=1}^{n} a_j}$ 的概率做操作 $a_i \leftarrow a_i + 1$。求 d 次操作后数组 a 的前 r 大值之和的期望，保留浮点数，误差不超过 10^{-6}。

$1 \le n, d \le 500, 1 \le r \le n$。

时间限制：3s；空间限制：256MB

3.3 知识考察

动态规划及优化、二项式反演

3.4 算法介绍

3.4.1 朴素动态规划算法及优化

做一个转化：每次操作有 $\sum_{i=1}^{n} a_i$ 种选择，其中 a_i 种操作是做 $a_i \leftarrow a_i + 1$ 的自增。这样可以将期望转化为权值和除以方案数。方案数容易计算，即 $\prod_{i=n}^{n+d-1} i$。那么要算的就是所有方案前 r 大和的和。

考虑从大到小加入数进行计算。为了下文描述方便，将 a_i 全部减去 1，最后答案加上 r。

对最终 a_i 取值进行动态规划。设 $f_{i,j,k}, g_{i,j,k}$ 分别表示考虑了最终的序列 a 中 $\ge i$ 的数，总共有 j 个 $\ge i$ 的数，它们的和为 k 的方案数和所有方案的前 $\min(j, r)$ 大和。初始值为 $f_{d+1,0,0} = 1$，其余为 0。

从大到小枚举 i, j, k 转移，枚举 l 表示序列中 $i-1$ 的数量，乘上 l 个 $i-1$ 在序列中选择位置的方案数、$l(i-1)$ 次操作在最终操作序列中选择位置的方案数，为 $c = \binom{n-j}{l}\binom{d-k}{i-1 \quad \dots \quad i-1 \quad d-k-l(i-1)}((i-1)!)^l$，其中多重组合里的省略号表示有 l 个 $i-1$。可以得到转移式

$$f_{i-1,j+l,k+l(i-1)} \leftarrow f_{i-1,j+l,k+l(i-1)} + f_{i,j,k}c$$

$$g_{i-1,j+l,k+l(i-1)} \leftarrow g_{i-1,j+l,k+l(i-1)} + (g_{i,j,k} + f_{i,j,k}\max(0, \min(l, r-j))(i-1))c$$

直接转移复杂度 $O(d^4)$ 无法接受。

注意到 $\binom{d-k}{i-1 \quad \dots \quad i-1 \quad d-k-l(i-1)}((i-1)!)^l = \frac{(d-k)!}{(d-k-l(i-1))!}$，而答案的 k 下标恰为 d，可以发现任一方案在动态规划的过程中选操作在最终操作序列的位置的系数的乘积恰好是 $d!$。这样可以在转移中省去多重组合和后面的 $((i-1)!)^l$ 的系数，而在最后将答案乘上 $d!$。为了方便描述下文算法均不在主要算法流程中计算选择操作在最终操作序列中的位置的方案数。

注意到该动态规划算法中有很多的冗余转移，如 $k + l(i-1) > d$ 的转移是冗余的。注意到转移有意义需要满足：$j \le \lfloor \frac{d}{i} \rfloor, ij \le k \le d, 0 \le l \le \lfloor \frac{d-k}{i-1} \rfloor$，那么有意义转移数总和为

$$\sum_{i=2}^{d+1}\sum_{j=0}^{\lfloor \frac{d}{i} \rfloor}\sum_{k=ij}^{d}\sum_{l=0}^{\lfloor \frac{d-k}{i-1} \rfloor} 1 \le \sum_{i=2}^{d+1}\sum_{j=0}^{\lfloor \frac{d}{i} \rfloor}\sum_{k=0}^{d}\sum_{l=0}^{\lfloor \frac{d}{i-1} \rfloor} 1 = O(d^3 \sum_{i=1}^{d} \frac{1}{i(i+1)}) = O(d^3 \sum_{i=1}^{d} \frac{1}{i} - \frac{1}{i+1}) = O(d^3)$$

那么只需要将枚举的上下界处理好，时间复杂度就是 $O(d^3)$ 的了。可以使用滚动数组优化空间。

3.4.2 基于差分计算贡献的动态规划算法

接下来的两个算法都围绕下式：设 a_i 是单调不增序列，那么

$$\sum_{i=1}^{r} a_i = \sum_{i=1}^{+\infty} \min(r, \sum_{j=1}^{n} [a_j \geq i])$$

考虑使用这一个式子以序列取值为状态设计动态规划：设 $f_{i,j,k}$ 和 $g_{i,j,k}$ 表示考虑了 $1 \sim i$ 的贡献，$\geq i$ 的数共有 j 个，$\sum_{p=1}^{n} \min(a_p, i) = k$ 的方案数和权值和。则枚举 $\geq i+1$ 的数共有 l 个，转移是

$$f_{i+1,l,k+l} \leftarrow f_{i+1,l,k+l} + f_{i,j,k} \binom{j}{l}$$

$$g_{i+1,l,k+l} \leftarrow g_{i+1,l,k+l} + (g_{i,j,k} + f_{i,j,k} \min(l, r)) \binom{j}{l}$$

直接做复杂度仍然是 $O(d^4)$，但 i 维度在转移和答案求解上都没有意义，故设 $f'_{j,k} = \sum_i f_{i,j,k}, g'_{j,k} = \sum_i g_{i,j,k}$，对 f', g' 进行同样的动态规划。相比于上文提到的朴素动态规划算法，这个算法更加简洁，复杂度仍然是 $O(d^3)$。

3.4.3 基于反演的算法

设 $W_{i,j}$ 表示序列 a 中恰有 i 个位置 $\geq j$ 的方案数，那么 $ans = \sum_{ij \leq d} W_{i,j} \min(i, r)$。
恰好不易计算，考虑算 $V_{i,j}$ 表示强制选择 i 个位置满足 $\geq j$ 的方案数，那么有 $V_{i,j} = \binom{n}{i}\binom{d-ij+n-1}{n-1}$。
而 $V_{i,j} = \sum_{k \geq i} \binom{k}{i} W_{k,j}$，由二项式反演有 $W_{i,j} = \sum_{k \geq i} (-1)^{k-i} \binom{k}{i} V_{k,j}$。将其代入答案式有

$$ans = \sum_{ij \leq d} W_{i,j} \min(i, r) = \sum_{ij \leq d} \min(i, r) \sum_{k \geq i, jk \leq d} (-1)^{k-i} \binom{k}{i} \binom{n}{k} \binom{d - jk + n - 1}{n - 1}$$

考虑有多少个合法的三元组 (i, j, k)，可以知道三元组个数为

$$\sum_{j=1}^{d} \sum_{i=1}^{\frac{d}{j}} \sum_{k=1}^{\frac{d}{j}} 1 = \sum_{j=1}^{d} (\frac{d}{j})^2 = O(d^2)$$

化简过程与朴素动态规划算法的优化类似。这样进行计算，复杂度为 $O(d^2)$。
对于这个算法，我们甚至可以通过化简并使用狄利克雷卷积进行优化做到 $O(d \log \log d)$。
但该算法的一个致命缺陷是精度误差较大，通过该题需要实现实数高精度。具体可见下一节。

3.4.4 对 C++ 自带浮点数变量类型计算精度的浅谈

注意到本题与在信息学竞赛中常见的计数问题有些不同，需要输出浮点数，这意味着答案不能取模。一个解决方式是使用高精度，但需要写一定量的代码，同时常数也会略大。此时可以考虑使用 C++ 自带的浮点数变量类型直接计算。但使用浮点数仍然有较大限制：

1. `double` 支持存储的数的范围的绝对值约在 10^{308} 范围内，超出这个值则会变成 `inf`。故储存时和运算中间过程中任意一个数的值不能超过它。在本题中的重要体现是计算组合数时直接储存阶乘是不可取的，而使用杨辉三角递推计算组合数是可取的。与此同时 `long double` 允许存储的绝对值范围约为 10^{4900}，可以存下阶乘。与此类似的如果小数点后位数过多会变成 0。

2. 浮点数的储存机制使得如果储存的值合法，那么就算其值域非常大或非常小，其最高的若干位可以被较精确地保留（`double / long double` 分别可以保证约 $15 \sim 16$ 和 $18 \sim 19$ 位的精度）。所以如果在浮点数运算中仅包含加法和乘除法，由于最后仅需要前若干位合法，直接运算不会出现问题；对于减法由于可能两个数本身的值较大而差却较小，这样会导致较大的精度误差。比如在本题中算法 4 就由于出现减法无法使用 C++ 自带浮点数变量通过本题，需要自行编写高精度。

更多关于浮点误差分析的内容可参考资料 [2]。

3.5 参考资料

[1] world final 2018 solution: http://www.csc.kth.se/~austrin/icpc/finals2018solutions.pdf
[2] 浮点数误差: https://zhuanlan.zhihu.com/p/133957594