2-SAT、欧拉路

李淳风

长郡中学

2024 年 9 月 29 日

2-SAT 问题

有 n 个变量，每个变量只有两种可能的取值。再给定 m 个条件，每个条件都是对两个变量的取值限制。求是否存在对 n 个变量的合法赋值，使 m 个条件均满足。这个问题被称为 2-SAT 问题，或者 “2-可满足性” 问题。

设一个变量 $A_i(1 \le i \le n)$ 的两种取值分别为 $A_{i,0}$ 和 $A_{i,1}$。在 2-SAT 问题中，M 个条件都可以转化为统一的形式—— “若变量 A_i 赋值为 $A_{i,p}$，则变量 A_j 必须赋值为 $A_{j,q}$”，其中 $p, q \in \{0, 1\}$。

2-SAT 问题的判定方法如下：

- 建立 $2N$ 个节点的有向图，每个变量 A_i 对用 2 个节点，一般设为 i 与 $i + n$。
- 考虑每个条件，形如 “若变量 A_i 赋值为 $A_{i,p}$，则变量 A_j 必须赋值为 $A_{j,q}$”，$p, q \in \{0, 1\}$。从 $i + p * n$ 向 $j + q * n$ 连一条有向边。

需要注意的是，上述条件蕴含它的逆否命题 “若变量 A_j 赋值为 $A_{j,1-q}$，则变量 A_i 必须赋值为 $A_{i,1-p}$”。如果该逆否命题没有在原命题中出现，我们应该把它对应的边也连上，从 $j + (1 - q) * n$ 向 $i + (1 - p) * n$ 连一条有向边。

总而言之，根据原命题和逆否命题的对称性，2-SAT 建出的有向图一定能画成 “一侧是节点 $1 \sim n$，另一侧是节点 $n + 1 \sim 2n$” 的形态。当把图中的边看作无向边时，这两边的情况时对称的。

2-SAT 问题

接着我们可以用 Tarjan 算法求出图中所有的强连通分量。若存在 $i \in [1,n]$，满足 i 和 $i+n$ 属于同一个强连通分量，则意味着：若变量 A_i 赋值为 $A_{i,p}$，则变量 A_i 必须赋值为 $A_{i,1-p}$ $\big(\forall p \in \{0,1\}\big)$。这显然时矛盾的，说明问题无解。若不存在这样的 i，则说明存在一组解。时间复杂度为 $O(n+m)$。

2-SAT 问题

接着我们可以用 Tarjan 算法求出图中所有的强连通分量。若存在 $i \in [1, n]$，满足 i 和 $i + n$ 属于同一个强连通分量，则意味着：若变量 A_i 赋值为 $A_{i,p}$，则变量 A_i 必须赋值为 $A_{i,1-p}$ $(\forall p \in \{0, 1\})$。这显然时矛盾的，说明问题无解。若不存在这样的 i，则说明存在一组解。
时间复杂度为 $O(n + m)$。
至于输出方案，我们可以发现，如果在有向图中从 i 出发能够到达 $i + n$，则说明 A_i 的取值应该为 $A_{i,1}$；若从 $i + n$ 出发能够到达 i，则说明 A_i 的取值应该为 $A_{i,0}$。因此我们只需要判断，它们在缩点后的图中拓扑序中的位置谁先谁后即可。再考虑 Tarjan 是从底向上来依次给 SCC 编号的，因此我们只需要比较两个节点的 SCC 编号，取 SCC 编号小的节点对应的取值即可。
与之前利用并查集处理二元关系比较，我们使用并查集的前提是关系具有传递性，并且关系是“无向”的。而在 2-SAT 问题中，关系是“有向”的，并且传递性也不用我们去强调，肯定是满足的。因此，2-SAT 问题符合二元关系的一般逻辑，能够处理更多、更复杂的问题。

例题

Katu Puzzle

有 n 个变量 $X_1 \sim X_n$，每个变量的取值是 0 或 1。给定 m 个算式，每个算式形如 $X_a\ op\ X_b = c$，其中 a, b 是两个变量的编号，c 是数字 0 或 1，op 是 and, or, xor 三个位运算之一。求是否存在对每个变量的合法赋值，使所有算式成立。
$1 \leq n \leq 1000, 1 \leq m \leq 10^6$。

例题

Katu Puzzle

有 n 个变量 $X_1 \sim X_n$，每个变量的取值是 0 或 1。给定 m 个算式，每个算式形如 $X_a\ op\ X_b = c$，其中 a, b 是两个变量的编号，c 是数字 0 或 1，op 是 and, or, xor 三个位运算之一。求是否存在对每个变量的合法赋值，使所有算式成立。
$1 \le n \le 1000, 1 \le m \le 10^6$。

如果只有异或运算，我们可以用带拓展域的并查集来解决该问题。现在有三种运算，我们先来尝试把它转化成 2-SAT 能接受的统一的形式。

Katu Puzzle

设节点 a 表示变量 X_a 赋值为 0，$a+n$ 表示 X_a 赋值为 1。

1. $a\ and\ b=0$：若 X_a 为 1，则 X_b 为 0；若 X_b 为 1，则 X_a 为 0。连接两条有向边 $(a+n,b),(b+n,a)$。
2. $a\ and\ b=0$：若 X_a 为 0，则 X_a 为 1；若 X_b 为 0，则 X_b 为 1。连接两条有向边 $(a,a+n),(b,b+n)$。
3. $a\ or\ b=1$：与第一种情况类似，连边 $(a,b+n),(b,a+n)$。
4. $a\ or\ b=0$：与第二种情况类似，连边 $(a+n,a),(b+n,b)$。
5. $a\ xor\ b=0$：连边 $(a,b),(b,a),(a+n,b+n),(b+n,a+n)$。
6. $a\ xor\ b=1$：连边 $(a,b+n),(b+n,a),(b,a+n),(a+n,b)$。

最后再通过 Tarjan 检查 a 和 $a+n$ 是否属于同一个 SCC 即可。

牧师约翰最忙碌的一天

牧师约翰在 9 月 1 日这天非常的忙碌。
有 N 对情侣在这天准备结婚，每对情侣都预先计划好了婚礼举办的时间，其中第 i 对情侣的婚礼从时刻 S_i 开始，到时刻 T_i 结束。
婚礼有一个必须的仪式：站在牧师面前聆听上帝的祝福。这个仪式要么在婚礼开始时举行，要么在结束时举行。
第 i 对情侣需要 D_i 分钟完成这个仪式，即必须选择 $S_i \sim S_i + D_i$ 或 $T_i - D_i \sim T_i$ 两个时间段之一。
牧师想知道他能否满足每场婚礼的要求，即给每对情侣安排 $S_i \sim S_i + D_i$ 或 $T_i - D_i \sim T_i$，使得这些仪式的时间段不重叠。
若能满足，还需要帮牧师求出任意一种具体方案。注意，约翰不能同时主持两场婚礼，且所有婚礼的仪式均发生在 9 月 1 日当天。如果一场仪式的结束时间与另一场仪式的开始时间相同，则不算重叠。
$1 \leq N \leq 1000$。

牧师约翰最忙碌的一天

牧师约翰在 9 月 1 日这天非常的忙碌。
有 N 对情侣在这天准备结婚，每对情侣都预先计划好了婚礼举办的时间，其中第 i 对情侣的婚礼从时刻 S_i 开始，到时刻 T_i 结束。
婚礼有一个必须的仪式：站在牧师面前聆听上帝的祝福。这个仪式要么在婚礼开始时举行，要么在结束时举行。
第 i 对情侣需要 D_i 分钟完成这个仪式，即必须选择 $S_i \sim S_i + D_i$ 或 $T_i - D_i \sim T_i$ 两个时间段之一。
牧师想知道他能否满足每场婚礼的要求，即给每对情侣安排 $S_i \sim S_i + D_i$ 或 $T_i - D_i \sim T_i$，使得这些仪式的时间段不重叠。
若能满足，还需要帮牧师求出任意一种具体方案。注意，约翰不能同时主持两场婚礼，且所有婚礼的仪式均发生在 9 月 1 日当天。如果一场仪式的结束时间与另一场仪式的开始时间相同，则不算重叠。
$1 \leq N \leq 1000$。

首先，每场婚礼是一个变量，有“开始时举行仪式”和“结束前举行仪式”两种取值。然后，我们可以 $O(N^2)$ 枚举没两场婚礼，判断一下它们的时间段是否会有冲突，转化为 2-SAT 模型并进行连边，注意逆否命题。最后，用 Tarjan 算法求 SCC，判断是否无解，有解的话最后输出方案即可。

欧拉路

最后，我们来探究一个关于无向图连通的小问题——欧拉路问题，俗称“一笔画”问题。

给定一张无向图，若存在一条从节点 S 到节点 T 的路径，恰好不重不漏地经过每条边一次（可以重复经过图中地节点），则称该路径为 S 到 T 地欧拉路。

特别地，若存在一条从节点 S 出发的路径，恰好不重不漏地经过每条边一次（可以重复经过图中地节点），最终回到起点 S，则称该路径为欧拉回路。存在欧拉回路的图被称为欧拉图。

欧拉图的判定也很简单，一张欧拉图为无向图，当且仅当无向图连通，并且每个点的度数都是偶数。欧拉路的判定几乎一样，只不过要求恰好有两个点的度数为奇数。这两个度数为奇数的节点就是欧拉路的起点 S 和终点 T。

欧拉路

判定无向图存在欧拉回路之后，如果想要输出方案，一个比较直接的想法是直接 DFS，对于每个节点 u 枚举从 u 出发的所有边，如果这条边之前还没有被遍历过，则打上标记并往下递归。因为每个点的度数都为偶数，所以一旦通过某条边到达了一个节点，就一定能通过另外一条边出去。
但需要注意到，这样实际上找出了若干条回路。我们还需要把这些回路按照适当的方法拼接在一起，形成整张图的欧拉回路。所以我们不存边，转而存点，在 DFS 的时候记录一个栈，如果当前节点 x 的所有边都已经遍历过了，那么就把 x 入栈。最后输出栈中所有元素，就找到了一条欧拉回路。
暴力做的复杂度是 $O(nm)$，但注意到我们在节点 x 时，是按顺序访问边的，我们可以不再从头开始枚举所有边，用数组记录第一条没有访问过的边，从这条边开始访问。这样就能把复杂度降低到 $O(n+m)$ 了。