二分图

李淳风

长郡中学

2024 年 9 月 30 日

前言

如果一张无向图的 N 个节点 $(N > 2)$ 可以分成 A, B 两个非空集合，其中 $A \cap B = \varnothing$，并且在同一集合内的点之间都没有边相连，那么称这张无向图为一张二分图，A, B 分别称为二分图的左部和右部。
因此，根据二分图的定义，我们知道如果两个集合中的点分别染成黑色和白色，可以发现二分图中的每一条边都一定是连接一个黑色点和一个白色点。并且二分图不存在长度为奇数的环。因为二分图的每一条边都是从一个集合走到另一个集合，只有走偶数次才能回到原本的集合。
根据这些性质，我们可以用染色法进行二分图的判定。大致思想为，尝试用黑白两种颜色标记图中的节点，当一个节点被标记后，它的所有相邻节点应该被标记成与它相反的颜色。若标记过程中产生冲突，则说明图中存在奇环。该过程一般基于深度优先遍历实现，时间复杂度为 $O(n + m)$。

例题

关押罪犯

S 城现有两座监狱，一共关押着 N 名罪犯，编号分别为 $1 \sim N$。他们之间的关系自然也极不和谐。很多罪犯之间甚至积怨已久，如果客观条件具备则随时可能爆发冲突。我们用“怨气值”（一个正整数值）来表示某两名罪犯之间的仇恨程度，怨气值越大，则这两名罪犯之间的积怨越多。如果两名怨气值为 c 的罪犯被关押在同一监狱，他们俩之间会发生摩擦，并造成影响力为 c 的冲突事件。

每年年末，警察局会将本年内监狱中的所有冲突事件按影响力从大到小排成一个列表，然后上报到 S 城 Z 市长那里。公务繁忙的 Z 市长只会去看列表中的第一个事件的影响力，如果影响很坏，他就会考虑撤换警察局长。

在详细考察了 N 名罪犯间的矛盾关系后，警察局长觉得压力巨大。他准备将罪犯们在两座监狱内重新分配，以求产生的冲突事件影响力都较小，从而保住自己的乌纱帽。假设只要处于同一监狱内的某两个罪犯间有仇恨，那么他们一定会在每年的某个时候发生摩擦。

那么，应如何分配罪犯，才能使 Z 市长看到的那个冲突事件的影响力最小？这个最小值是多少？

$N \leq 20000, M \leq 100000$

关押罪犯

我们在之前已经用并查集解决过了这个问题，此处我们尝试用二分图再次进行求解。
首先，我们发现答案具有单调性。因此我们可以二分答案 mid，转化为一个判定问题，判断是否存在一种分配罪犯的方案，使 Z 市长看到的那个冲突事件的影响力不超过 mid。
那么，对于任意两个仇恨程度大于 mid 的罪犯都必须被安排在不同的监狱。我们把罪犯作为节点，在仇恨程度大于 mid 的两个罪犯之间连边，得到一张无向图。这张无向图中的节点需要被分成两个集合（两个监狱），每个集合内部的点之间没有边（不能发生影响力大于 mid 的事件），因此，我们用染色法判断这张图是否是二分图即可。

二分图最大匹配

“任意两条边都没有公共端点”的边的集合被称为图的一组匹配。在二分图中，包含边数最多的一种匹配被称为二分图的最大匹配。
对于任意一组匹配 S（S 是一个边集），属于 S 的边被称为“匹配边”，“不属于 S 的边被称为非匹配边”，“匹配边”的端点被称为“匹配点”。如果在二分图中存在一条连接两个非匹配点的路径 $path$，使得非匹配边与匹配边在 $path$ 上交替出现，那么称 $path$ 是匹配 S 的增广路，也称交错路。
增广路显然有以下性质：

1. 长度 len 是奇数。
2. 路径上第 $1, 3, 5, \cdots, len$ 条边是非匹配边，第 $2, 4, 6, \cdots, len-1$ 条边是匹配边。

所以如果我们把路径上的所有边的状态取反，原来的匹配边变为非匹配边，原来的非匹配边变成匹配边，那么新的到的边集 S' 仍然是一组匹配，并且匹配边数增加了 1。进一步可以得到推论：
二分图的一组匹配 S 是最大匹配，当且仅当图中不存在 S 的增广路。

匈牙利算法（增广路算法）

匈牙利算法，又称增广路算法，用于计算二分图最大匹配。它的主要过程为：

1. 设 $S = \varnothing$，即所有的边都是非匹配边；
2. 寻找增广路 $path$，把路径上所有边的匹配状态取反，得到一个更大的匹配 S'；
3. 重复第二步，直至图中不存在增广路。

该算法的关键在于如何找到一条增广路。匈牙利算法依次尝试给每一个左部节点 x 寻找一个匹配的右部节点 y。右部节点 y 能够和左部节点 x 匹配，需要满足以下两个条件之一：

1. y 本身就是非匹配点，这时边 (x, y) 自己就构成一条增广路。
2. y 已经与左部点 x' 匹配，但从 x' 出发能找到一条增广路。

在实际的程序实现中，我们采用深度优先搜索的框架，递归地从 x 出发寻找增广路。若找到，则在深搜回溯中，正好把路径上的匹配状态取反。另外，可以用全局数组记录每个点的访问情况，避免重复搜索。

匈牙利算法（增广路算法）

匈牙利算法的正确性基于贪心策略，它的一个重要特点是：当一个点成为匹配点后，至多因为找到增广路而更换对象，但绝对不会变回非匹配点。
对于每个左部节点，寻找增广路至多遍历整张二分图一次，因此该算法的复杂度为 $O(nm)$。

```
bool dfs(int x,int cnt){
    for(int i=head[x];i;i=Next[i])
        if(vis[y=ver[i]]!=cnt){//vis 数组为 int 类型
            vis[y]=cnt;
            if(!match[y] || dfs(match[y],cnt)){
                match[y]=x; return 1;
            }
        }
    return 0;
}

for(int i=1;i<=n;i++)//在 main 函数中
    if(dfs(i,i)) ans++;//这样可以不需要每次清空 vis 数组
```

例题

棋盘覆盖

给定一个 n 行 m 列的棋盘，有一些格子禁止放置。求最多能不重叠地往棋盘上放多少块长度为 2 、宽度为 1 的骨牌。

$n, m \leq 100$。

例题

棋盘覆盖

给定一个 n 行 m 列的棋盘，有一些格子禁止放置。求最多能不重叠地往棋盘上放多少块长度为 2 、宽度为 1 的骨牌。
$n, m \leq 100$。

在之前的学习中，我们用状态压缩动态规划来解决了用 $1 * 2$ 的长方形铺满 $n * m$ 网格的方案数问题。与之不同的是，本题只要求能防止骨牌的最大数量，并且有站感悟。当 n, m 较小时，仍然能用状态压缩动态规划来解决，但更高效的方法则是构建出二分图匹配的模型。
二分图匹配的模型有两个要素：

1. 节点能分成独立的两个集合，每个集合内部有 0 条边；
2. 每个节点只能与一条匹配边相连。

在本题中，任意两张骨牌不能重叠，也就是每个格子只能被 1 张骨牌覆盖，这就满足了第二个条件；而如果把棋盘按照行数加列数的奇偶性黑白染色，那么两个相同颜色的格子之间没有边相连，这就满足了第一个条件。
我们把白色格子和黑色格子看作点，每次如果能用一张骨牌覆盖两个格子，就给对应的两个格子连一条无向边。最后的答案就是这个二分图的最大匹配，时间复杂度为 $O(n^2 m^2)$。

例题

棋盘覆盖

给定一个 N 行 M 列的棋盘，已知某些格子禁止放置。
问棋盘上最多能放多少个不能互相攻击的車。
車放在格子里，攻击范围与中国象棋的“車”一致。
$1 \leq N, M \leq 200$

例题

棋盘覆盖

给定一个 N 行 M 列的棋盘，已知某些格子禁止放置。
问棋盘上最多能放多少个不能互相攻击的車。
車放在格子里，攻击范围与中国象棋的“車”一致。
$1 \leq N, M \leq 200$

我们仿照上一道题，提取出两个条件：

1. 每行每列只能放一个车。如果在 (i, j) 放了车，就等于占用了第 i 行与第 j 列放车的“名额”。
2. 每个车显然不能既在第 i_1 行又在第 i_2 行，所以两个“行”对应的节点没有边。对于“列”同理。

因此，我们可以把行、列看作节点，一共有 $(n+m)$ 个节点。如果格子 (i, j) 没有被禁止，就在第 i 行、第 j 列对应的两个节点之间连边。
上述二分图的最大匹配数就是答案。

完备匹配与多重匹配

给定一张二分图，其左部、右部节点数相同，均为 N 个节点。如果该二分图的最大匹配包含 N 条匹配边，则称该二分图具有完备匹配。

给定一张包含 N 个左部节点、M 个右部节点的二分图。从中选出尽量多的边，使得第 i 个左部节点至多与 kl_i 条选出的边相连，第 j 个右部节点至多与 kr_j 条选出的边相连。该问题被称为二分图的多重匹配。
多重匹配一般有下列解决方案：

1. 拆点，把第 i 个左部节点拆成 kl_i 个不同的左部节点，第 j 个右部节点拆成 k_j 个右部节点。对于每条原图中的边 (i, j)，在所有 i 拆成的节点与所有 j 拆成的节点之间连边。
2. 如果所有的 $kr_j = 1$，可以直接在匈牙利算法中，让每个左部节点执行 kl_i 次 dfs。
3. 如果所有的 $kl_i = 1$，可以修改匈牙利算法的实现，让右部节点可以匹配 kr_j 次，超过匹配次数后，依次尝试递归当前匹配的 kr_j 个左部节点，看能否找到增广路。
4. 网络流。这是最一般也最高效的解决方法。

例题

导弹防御塔

Freda 的城堡遭受了 M 个入侵者的攻击！
Freda 控制着 N 座导弹防御塔，每座塔都有足够数量的导弹，但是每次只能发射一枚。
在发射导弹时，导弹需要 T_1 秒才能从防御塔中射出，而在发射导弹后，发射这枚导弹的防御塔需要 T_2 分钟来冷却。
所有导弹都有相同的匀速飞行速度 V，并且会沿着距离最短的路径去打击目标。
计算防御塔到目标的距离 $Distance$ 时，你只需要计算水平距离，而忽略导弹飞行的高度。
导弹在空中飞行的时间就是 $Distance/V$ 分钟，导弹到达目标后可以立即将它击毁。
现在，给出 N 座导弹防御塔的坐标，M 个入侵者的坐标，T_1, T_2 和 V。
因为 Freda 的小伙伴 Rainbow 就要来拜访城堡了，你需要求出至少多少分钟才能击退所有的入侵者。
$N, M \leq 50$，坐标绝对值不超过 10000，T_1, T_2, V 为不超过 2000 的正整数。

导弹防御塔

首先，如果在一个较短的时间内能够击退所有入侵者，那么对于更长的时间显然也没有问题。所以我们可以先二分答案，转化为一个判定问题。
首先，每个入侵者被攻击 1 次之后就会被击退，每座塔虽然能够发射多枚导弹，但是每枚导弹只能攻击一个入侵者。而且，塔当然不可能去攻击塔，所以我们考虑把塔和入侵者看作节点，在它们之间连边。
现在已知发射预热时间 T_1、冷却时间 T_2，我们可以计算出每座防御塔在 mid 分钟内至多能够发射出的导弹数量，记为 P。这是一个多重匹配问题，我们需要把每座防御塔拆成 P 个节点，第 i 个节点往第 i 枚导弹能在 mid 分钟内消灭的所有入侵者节点连边。
用匈牙利算法计算二分图的最大匹配。如果每个敌人对应的节点都能找到匹配，则说明 mid 分钟内能击退所有入侵者。