04.1 - Testing for bipartiteness

Bipartite Graphs 二分图

一个图 $G=(V,E)$ 是二分图，当且仅当其顶点集 V 可以划分为两个不相交的子集 A 和 B，使得图中的每条边的两个端点分别属于 A 和 B

一般，我们将其写为 $G=(A\cup B,E)$

其他定义：

- 一个图 $G=(V,E)$ 是二分图，当且仅当可以用两种颜色（例如红色和绿色）对其节点进行着色，使得每条边的两个端点分别有不同的颜色
- 一个图 $G=(V,E)$ 是二分图，当且仅当它不包含任何奇数长度的环

这些替代定义有时也被称为“特征描述”，因为它们提供了对二分图性质的不同视角理解和描述

奇数环

一个图 $G=(V,E)$ 是二分图，当且仅当它不包含任何奇数长度的环

证明:
假设 G 是二分图，其包含一个奇数环 $C=u_1,u_2,\dots,u_n,u$，其中某个顶点 $u\in B$

因为 G 是二分图，$u_1\in A,u_2\in B,u_3\in A,\dots$
这就是说 $\forall k\in\{1,2,\dots,n\}$：如果 k 是奇数，则 $u_k\in A$；如果 k 是偶数，则 $u_k\in B$

由于 C 是奇数环，因此 $u_n\in B$，因此 $u\in A$，这与假设矛盾，因此得证

BFS 检测二分图

- 使用BFS遍历图中的每一个节点，按层级进行遍历：如果一个节点在第1层，则将其标记为红色；如果在第2层，则将其标记为绿色
- 在BFS过程中，根据当前节点所在的层级是奇数还是偶数来决定节点的颜色
- 最后，检查所有的边，确保没有边的两个端点具有相同的颜色

```
Algorithm BFS(G, s)
    Initialise empty list L0
    Initialise colour list C
    Insert s into L0
    Set C[s] = red

    Set i = 0
    While Li is not empty
        Initialise empty list Li+1
        for each node v in Li
            for all edges e incident to v
                if edge e is unexplored
                    let w be the other endpoint of e
                    if node w is unexplored
                        label e as discovery edge
```

```
                        insert w into Li+1
                        If i+1 is odd, set C[w] = green, else set C[w] = red
                    else
                        label e as cross edge
        i = i + 1

    For all edges e = (u, v) in G
        If C[u] = C[v] return "not bipartite"
    Return "bipartite"
```

1. 初始化空列表 L_0 用于存储当前层的节点
2. 初始化颜色列表 C 用于存储每个节点的颜色
3. 将起始节点 s 插入 L_0 并将其颜色设置为红色
4. 设置层级计数器 $i = 0$
5. 当当前层的列表 L_i 不为空时，执行以下操作：
    1. 初始化空列表 L_{i+1} 用于存储下一层的节点
    2. 对于当前层的每个节点 v，遍历所有与 v 相连的边 e：
        1. 如果边 e 未被探索：
            1. 获取边 e 的另一个端点 w
            2. 如果节点 w 未被探索：
                1. 将边 e 标记为发现边
                2. 将节点 w 插入 L_{i+1}
                3. 如果 $i + 1$ 是奇数
                    1. 则将 w 设置为绿色
                4. 否则
                    1. 将 w 设置为红色
            3. 否则
                1. 将边 e 标记为交叉边
    3. 层级计数器 i 增加 1
6. 遍历图中所有边 $e = (u, v)$
    1. 如果 u 和 v 的颜色相同
        1. 返回 "not bipartite"
    2. 如果没有颜色冲突
        1. 返回 "bipartite"

运行时间

我们在 BFS 检测二分图中：

- 为起始节点分配颜色
- 在BFS循环中，对每个节点进行奇偶检查，并根据结果分配颜色
- 在BFS结束后，增加一个额外的循环，检查图中每条边的两个端点的颜色，确保它们不同

这些额外操作不会显著增加算法的复杂度，因为每个操作的复杂度都在BFS的基本复杂度范围内

因此，这个算法的复杂度是 $O(n + m)$