05.2 - Finding Strongly Connected Components

连通分量

- 连通分量（Connected Component）：在无向图 G 中，连通分量是指任意两个节点通过某条路径可以连接的子图
- 强连通分量（Strongly Connected Component）：在有向图 G 中，强连通分量是指任意两个节点可以相互到达的子图

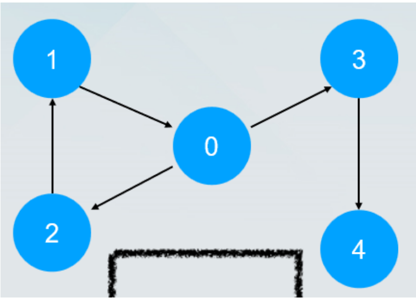
强连通分量（Strongly Connected Components, SCC）

如何在图 G 中找到所有的强连通分量?

我们可以为一个节点 s 运行 forward 和 backward 的 BFS，找到从节点 s 可以相互到达的节点集合，这个集合就是 s 的强连通分量
BFS可能会根据我们访问节点的方式产生不同的连通分量。因此，我们需要在 forward 和 backward 遍历时使用一致的访问方式

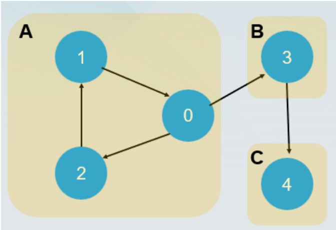
Kosajaru's Algorithm

1. 在 G 上从任意一个节点 s 开始，执行 DFS
2. 将 DFS 遍历的节点添加到栈中
   - 当某个节点的 DFS 遍历完成时，将该节点添加到栈中
3. 按照从栈中弹出节点的顺序，在 G^{rev} 中执行 DFS
4. 第二次 DFS 产生的 DFS 树就是强连通分量



1. 对于这个图，我们从任意一个节点开始，比如 0
2. 执行 DFS：
   1. 0 有两个出边，随机选一条，比如 $(0, 3)$
      1. 这条路线是 $0 \to 3 \to 4$，把 4 放到栈底，3 放在 4 上面
   2. 选择另一条路线
      1. 这条路线是 $0 \to 2 \to 1$，把 1 放到 3 上面，2 放到 3 上面
   3. 最后把 0 放到最上面
      1. 我们得到这样的栈：$0, 2, 1, 3, 4$（左边在上）
3. 将图的方向反转，从栈顶到栈底执行 DFS
   1. 栈顶是 0，执行 DFS，得到枝条 $0 \to 1 \to 2$
   2. 2 和 3 已经在其中，跳过
   3. 对 3 执行 DFS，得到枝条 3

    4. 对 4 执行 DFS，得到枝条 4
4. 输出第三步得到的 DFS tree，这就是强连通分量（0210, 3, 4）



时间复杂度
进行了两次 DFS，一次 DFS 是 $O(n+m)$，两次也是这个

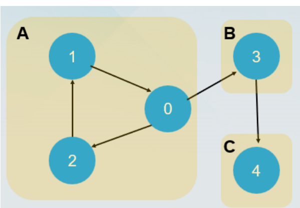
强连通分量之间的无路径性引理（Lemma of No Path between Distinct Strongly Connected Components）

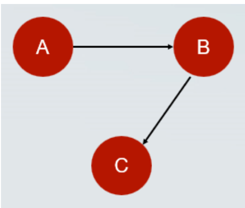
- 设 C 和 C' 是有向图 G 中的两个不同的 SCC
    - 设 u, v 属于 C，u', v' 属于 C'
- 假设 G 中存在从 u 到 u' 的路径 (1)
- 那么 G 中不可能存在从 v' 到 v 的路径

证明：
假设存在从 v' 到 v 的路径 (2)
因为 u' 和 v' 在同一个 SCC 中，因此他们之间存在路径
从 (1) 与上一步，存在从 u 到 v' 的路径
所以存在一个从 v' 到达 u 的路径
这说明 u 与 v' 相互可达
这与 C 和 C' 是两个不同的 SCC 矛盾

分量图（Component Graph）

分量图（Component Graph），也称为缩点图（Condensation Graph），是在图论中将一个有向图的强连通分量作为单个节点处理而得到的新的有向图



⇒


对于两个不同的连通分量，从第一个到第二个可以有一条路径，或者反之，但不能两者都有

分量图是一个有向无环图（DAG）

Lemma

- 设 C 和 C' 是图 G 中两个不同的强连通分量
- 假设存在一条有向边跨越 C 和 C'
- 那么，对 C 中节点执行的 DFS 会在对 C' 中节点执行的 DFS 之后完成

Corollary
如果前向 DFS 在分量 C 上完成的时间晚于在分量 C' 完成的时间，那么：

- 在图 G 中不存在从 C' 到 C 的跨越边
- 在反向图 G^{rev} 中不存在从 C 到 C' 的跨越边

这意味着在反向图 G^{rev} 上的后向 DFS 中，如果我们从在前向 DFS 中最后完成的 SCC 开始，我们将不会找到指向其他 SCC 的边

（上面的证明太简单了，我不写了）

后向 DFS 会按照拓扑排序访问强连通分量图 G^{SCC} 的节点
Alt view：
生成 G^{SCC} 的拓扑排序
在考虑 SCC 上 G^{rev} 上按照这个拓扑排序执行 DFS