hw2 刘佳璇 231220105

第八章

8.1 有向图DFS点与边的着色

1）

边 uv 为TE：点 v 可能是白色。

边 uv 为BE：点 v 可能是灰色。

边 uv 为DE：点 v 可能是黑色。

边 uv 为CE：点 v 可能是黑色。

2）

点 v 为白色：边 uv 可能是TE。

点 v 为灰色：边 uv 可能是BE。

点 v 为黑色：边 uv 可能是DE、CE。

3）

点 v 为白色当且仅当边 uv 为TE。

充分性：若点 v 为白色，点 v 会被点 u 访问。由定义，边 uv 为TE。

必要性：若边 uv 为TE，点 v 第一次被访问，因此点 v 为白色。

点 v 为灰色当且仅当边 uv 为BE。

充分性：若点 v 为灰色，说明对点 v 的遍历尚未结束，即点 v 为点 u 的祖先节点，可知边 uv 为BE。

必要性：若边 uv 为BE，说明点 v 是点 u 在遍历中的祖先节点，此时对点 v 的遍历尚未结束，可知点 v 为灰色。

8.3 活动区间等价刻画DFS的过程

定理 8.1

1）

充分性：若 w 是 v 在DFS树中的后继节点，则 v 的活动区间在 w 的活动区间开始之前开始，且在其活动区间结束之后结束，即 $active(w) \subseteq active(v)$ 。且仅当 $w = v$ 时， $active(w) = active(v)$ 。

必要性：若 $active(w) \subseteq active(v)$ ，则遍历到 w 时已经遍历到 v ，且 v 未返回，即 w 是 v 在DFS树中的后继节点。

2）该命题的充分性和必要性与 1）的必要性和充分性分别为逆否命题。故正确。

3）

①

充分性： vw 是CE，说明当遍历 v 时， w 是黑色节点。因此 $active(w)$ 在 $active(v)$ 前面。

必要性： $active(w)$ 在 $active(v)$ 前面，说明当遍历 v 时， w 是黑色节点。因此 vw 是CE。

②

充分性： vw 是DE当且仅当存在中间节点 x ，满足 v 是 x 的祖先，且 x 是 w 的祖先。因此 $active(v) \subset active(x) \subset active(w)$ 。

必要性：同理。

③

充分性： vw 是TE，则 $active(v) \subset active(w)$ 。若存在 x 使得 $active(v) \subset active(x) \subset active(w)$ ，则 vw 是DE，矛盾。

必要性： $active(v) \subset active(w)$ ，则 w 是 v 的后继节点。又因为不存在 x 使得 $active(v) \subset active(x) \subset active(w)$ ，故 vw 不是DE。因此vw 是TE。

④

充分性： vw 是BE，则 w 是 v 的祖先节点，因此 $active(v) \subset active(w)$ 。

必要性：同理。

8.5 割点基于路径的定义

充分性：若点 v 为割点，则 $G' = G - \{v\}$ 不连通。在 G' 的连通片 G_1 中取一点 w ，连通片 G_2 中取一点 x ，若 w 和 x 间仍有路径不经过点 v ，则 G_1 和 G_2 连通，矛盾。

必要性：若 w 到 x 的所有路径上都经过点 v ，则删去 v 后 w 和 x 不连通，即 G 不连通。由割点定义知，点 v 为割点。

8.7 有向图的收缩图无环

证明：

采用反证法。假设图 G 的收缩图 $G\downarrow$ 有环 $v_1 \to v_2 \to \cdots \to v_n \to v_1$ ，则 v_i $(\forall\ 1 \leq i \leq n)$ 中的点可以通过路径 $v_i, v_{i+1}, \ldots, v_{j-1}, v_j$ 到达 v_j $(\forall\ 1 \leq j \leq n$ 且 $j \neq i)$ ，则 $\{v_1, v_2, \ldots, v_n\}$ 是强连通片，会被收缩为一点，矛盾。故有向图的收缩图无环。

8.8 用BFS实现强连通片算法

第一次DFS不可以替换。SCC算法需要实现强连通片之间的拓扑排序，并保证第一次DFS的源头在第二次DFS中是尽头。BFS依赖的队列结构不能标记源头和尽头，也不能实现拓扑排序。

第二次DFS可以替换。只需要保证能够遍历强连通片。

8.9 无向图的DFS树

点 v 是割点当且仅当点 v 存在至少两个直接子节点。

充分性：若点 v 为割点，则点 v 一定有子节点；如果点 v 只有一个子节点，则删去点 v ，图仍然连通。

必要性：若删去点 v 后，子节点 x 和 y 不连通，则 x 和 y 不在同一子树中，点 v 是割点。

8.10 寻找割点算法的 $back$

仍然正确。因为 $back$ 的值被更新时只会减小，所以只需要大于等于 $v.discoverTIme$ 。

8.11 证明定理的正确性

引理 8.9

充分性：因为边 uv 是桥，所以 u 的祖先节点（包括 u ）和 v 的后继节点之间的所有路径都经过边 uv ，即以 v 为根的所有遍历子树中没有BE指向 v 的祖先。

必要性：因为以 v 为根的所有遍历子树中没有BE指向 v 的祖先，所以去掉 uv 后， v 的后继节点与 v 的祖先节点不再相连通，所以边 uv 是桥。

定理8.6

当遍历从边 uv 回退时，如果 $v.back > u.discoverTime$ ，则由引理 8.9知，以 v 为根的子树中没有BE指向 v 的祖先，即边 uv 是桥；反之边 uv 不是桥。

8.14 无向图定向1

首先检查无向图的每个点是否都在环中，如果有点不在环中，则无法构造。

在环中任取一点 v ，以 v 为顶点进行DFS，规定边的方向与DFS的方向相同。

算法的时间复杂度为 $\mathrm{O}(n+m)$ 。

8.15 无向图定向2

1）

如果 G 中存在桥 uv ，则 v 的后继节点没有路径指向 u 的祖先节点， G 中不存在 SCC 定向。

2）

如果 G 中不存在桥，则任选一个点 r 作为顶点进行DFS，规定边的方向与DFS的方向相同。考察 r 的后继节点 u ，因为没有桥，所以以 u 为根的所有遍历子树中有BE指向 u 的祖先 v 。对 v 进行迭代，最终得到以 r 的子节点为根的所有遍历子树中有BE指向 r ，即得到强连通片。

算法的时间复杂度为 $\mathrm{O}(n+m)$ 。

8.18 寻找包含边 e 的环

假设边 e 的端点为 u 和 v ，若 u 和 v 不存在除彼此以外的其他邻居，则环一定不存在。

删除边 e 后以 u 为顶点进行DFS，如果能到达 v ，则存在环；反之不存在。

算法的时间复杂度为 $\mathrm{O}(n+m)$ 。

8.19 拓扑排序

```
stack s, result;
foreach node v in G do
    if v.in_degree=0 do
        s.push(v);
while !S.empty do
    u = s.pop();
    result.push(u);
    foreach neighbor w of u do
        w.in_degree--;
        if w.in_degree=0 do
            s.push(w);
return result;
```

如果存在环，环上的点不在 $result$ 中。

8.20 顶点间的“one-to-all”可达性问题

1）

以 s 为顶点进行DFS，如果DFS结束后还有白色节点，则不能到达图中其他所有点；反之可达。

2）

先进行 SCC 算法得到收缩图，如果收缩图中有两个及以上入度为 0 的点，则不存在“one-to-all”可达的节点；如果收缩图中没有入度为 0 的点，则每个节点都“one-to-all”可达；如果收缩图中有一个入度为 0 的点，则以该点为顶点进行DFS，如果DFS结束后没有白色节点，则该节点为“one-to-all”可达的节点，反之不存在。

8.22 影响力

1）

先进行 SCC 算法得到收缩图，对所有出度为 0 的强连通片，比较它们内部的节点个数，个数最少的强连通片中的点即为影响力最小的点。

2）

先进行 SCC 算法得到收缩图，统计所有强连通片内部节点的个数和它们能达到的强连通片，强连通片自身和它们能达到的强连通片的节点个数之和即为该强连通片内每个节点的影响力。找到其中最大值，即为所求。

8.24 选课

对图 G 进行拓扑排序，然后寻找关键路径。关键路径长度即为所求。

8.28 $2-SAT$ 问题

TODO

第九章

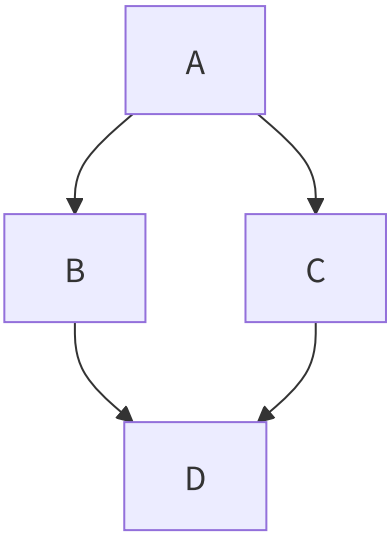
9.1 无向图BFS不存在BE和DE

BE：当遍历点 u 时，如果发现BE指向点 v ，则在遍历点 v 的子节点时已经访问过点 u ，BE是二次遍历，排除。

DE：当遍历点 u 时，如果发现DE指向点 v ，则在遍历点 v 的子节点时已经访问过点 u ，DE是二次遍历，排除。

9.2 BFS的白色路径定理

不成立。如图，刚发现1时，有白色路径2->4，但4的祖先可能是3。



9.3 DFS判断二分图

可以。只需要寻找长度为奇数的环。

DFS适合解决有关“尽头”的问题，BFS适合解决对所有邻居做相同处理的问题。

9.4 检测环

DFS：遇到BE则存在环。

BFS：有向图寻找BE，即找到黑色节点后判断该节点是否为当前节点祖先；无向图寻找CE，即灰色节点。

9.5 移除无向图的一条边使其保持连通

原命题等价于无向图中是否存在环，同 9.4 。

9.8 最小生成树

1）

采用DFS或BFS。

2）

先采用DFS或BFS生成一个生成树，然后遍历剩下11条边，对每条边 uv ，如果生成树上从 u 到 v 的路径中权值最大的边比 uv 的权值大，则用边 uv 替换。

3）

对所有权为 1 的边，用DFS或BFS遍历生成森林，然后将森林里的树看作点，用权为 2 的边生成新的生成树。

9.10 最短路径数量

TODO