算法习题课3

2023.5.17 朱俊俊

8.1

(1) TE->白色 BE->灰色 CE、DE->黑色

(2) 白色->TE 灰色->BE 黑色->CE、DE

(3) uv为TE当且仅当v为白色。

证明：=> uv为TE，说明DFS中存在u->v的访问，所以v为白色。

<= v为白色，说明u会访问v，因此uv为TE。

uv为BE当且仅当v为灰色。

证明：=> uv为BE，说明v是u的祖先，子孙未返回说明祖先也未返回，因此v为灰色。

<= v为灰色，说明v的dfs尚未返回，且v已被访问到，故v是u的祖先，因此uv为BE。

8.3

1) w是v在DFS树中的后继节点，当且仅当active(w) $\subseteq$ active(v)。若w≠v，此处为真包含。

证明：=>：w是v在DFS树中的后继节点，则遍历w时已遍历到v且v仍未返回，因此active(w) $\subseteq$ active(v)。

<=：反之同理。

2）w和v没有祖先后继关系，当且仅当active(w)和active(v)没有重叠。

证明：=>：显然无重叠。

<=：不妨设w是v的后代，则active(w) $\subseteq$ active(v)，但两者无重叠，矛盾。

3）

1．vw是CE，当且仅当active(w)在active(v)前面。

证明：vw是CE，说明w已是黑色节点，正在遍历v，因此active(w)在active(v)前面。反之同理。

2．vw是DE，当且仅当存在第三个节点x满足：active(w) $\subset$ active(x) $\subset$ active(v)。

证明：vw是DE，则存在存在中间节点x，v是x祖先，x是w祖先，故有active(w) $\subset$ active(x) $\subset$ active(v)。反之同理。

3．vw是TE，当且仅当active(w) $\subset$ active(v)，且不存在第三个节点x满足active(w) $\subset$ active(x) $\subset$ active(v)。

证明：v是w祖先，因此active(w) $\subset$ active(v)，若存在这样的x，则vw是DE，矛盾。反之同理。

4．vw是BE，当且仅当active(v) $\subset$ active(w)。

证明：vw是BE，因此w是v祖先，故active(v) $\subset$ active(w)。反之同理。

8.5

点v是割点，当且仅当存在点对w和x满足点v出现在w到x的所有路径上。

=>： v 是割点， $G' = G/\{v\}$ ， G' 不连通，则 G 中存在两点 w 和 x ，在 G 中连通，在 G' 中不连通。假设有一条 w 到 v 的路径，使得其不经过 v ，则在 G' 这条路径仍然存在， w 和 x 仍然连通，矛盾。

<=：若 w 到 x 所有路径上都经过了点 v ，则删去点 v 后 w 和 x 不连通，即 G 不连通，所以 v 是割点。

8.7

证明：有向图的收缩图是无环的。

记有向图 G 的收缩图为 G' 。

假设 G' 有环，则存在两个不同点 x 和 y ， x 和 y 相互到达。

即在 G 中，存在 $u \in x$ 和 $v \in y$ ， u 和 v 相互到达，则 u 和 v 必属于一个强连通片，矛盾。

8.8

强连通片的两次DFS能不能换成BFS?

- 第一次DFS不能换BFS。DFS是在顶点所在子树全部遍历完成后再进栈，首节点的活动区间包含同一个强连通片所有其他节点的活动区间。而BFS是在顶点的孩子进队列后就进栈，此时该顶点的孩子还没有进栈，不满足推论8.1。
- 第二次DFS可以换BFS。只要保证能遍历到顶点就行。

8.9

无向连通图的深度优先遍历树的根节点 v 是割点 <=> v 至少有2个子节点 x 和 y 。

=>： v 是割点，如果没有子节点，则 v 不是割点，如果只有1个子节点，那么删去 v 图还是连通的。

<=：若删去 v 后， x 和 y 不连通（若仍连通，则 x 和 y 应该在同一子树中）， v 是割点。

8.10

仍然正确。

back的初始化值只要充分大（大于等于于v.discoverTime）即可。

因为算法中back表示该节点所能联通的最先被访问的点，back的更改只会减小。

证明方法仍然是定理8.5

8.11

- 引理8.9 遍历树中的TE边uv为桥，当且仅当以v的根的所有遍历树子树中没有BE指向v的祖先。

  =>：假设有BE指向 v 的祖先，则删去 uv 后图仍连通，

  uv 不是桥，矛盾。

  <=：删去 uv 后， v 为根的子树无法到达 v 的祖先，图不连通，所以 uv 是桥。

- 定理8.6 BRIDGE-DFS算法是正确的。

  证明方法和定理8.5类似。

8.14

设计算法，判定是否可以为无向图G中的边添加方向，使得每个顶点入度至少为1。

- 找环，就是找BE，找不到则无解。
- 找到BE后，以BE任意端点作为顶点进行DFS，遍历边时进行定向，方向与遍历方向相同。
- 复杂度 $O(n+m)$

注意是无向图，找BE要看是不是当前边的反向边！

8.15

SCC定向：对无向连通图G的每条边确定方向，使得定向后的有向图强连通。

若G中存在桥，则G不存在SCC定向。若G中存在桥，线性时间给出G的一种SCC定向。

- 若G中有桥 uv ，则记删去 uv 后形成的两个连通分量 x 和 y ，无论 uv 如何定向，也无法保证 x 和 y 中的点能相互到达。
- 从任意节点开始DFS，定向的方向与边的方向相同。

8.18

线性时间判断无向图G中是否存在包含e的环。

设边e的顶点为u、v，暂时删去这条边e，然后从顶点u对修改后的图进行一次dfs，看能否到达点v。若能到达则图中有两条连通u-v的路径，因此有环，反之无环。

时间复杂度O(m+n)。

8.19

拓扑排序，重复地寻找入度为0的节点并输出，然后将该节点和其关联的出边删除。

```
TOPO()
queue q;
for each node u do
  if u.indeg == 0 then
    q.push(u)
while !q.empty() do
  u = q.front()
  result.push(u)
  q.pop()
  for each neighbor v of u do
    v.indeg--;
    if v.indeg == 0:
      q.push(v)
return result
```

如果有回路，则回路上的点不在 $result$ 中。

8.20

one-to-all 一个顶点能否到达所有顶点。是否存在一个节点可以到达所有顶点。

- 直接DFS，若有没有搜到的点，则不能到达图中其他所有节点。
- scc缩点，找到入度为0的点，从该点开始DFS，若有没有搜到的点，则不能到达图中其他所有节点。

8.22

影响力impact(v)为从v可达的顶点个数，找出影响力最小和最大的点。

- 缩点
- 出度为0的强连通片之间比较强连通片点的个数，最小的即是影响力最小的点。
- 入度为0的强连通片进行DFS，统计每个强连通片能搜到的顶点个数，找到最大的即是影响力最大的点。

8.24

构建有向图，课程为顶点，若u是v的先修课，建一条有向边uv。

DAG图寻找关键路径，dp。

$$f[u] = max(f[v]) + 1, (u, v) \in E$$

8.28

2SAT问题

（1）寻找满足赋值。

$x_1 = TRUE, x_2 = TRUE, x_3 = FALSE, x_4 = TRUE$

（2）构造不存在满足赋值的实例。

答案不唯一。

$(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (x_3 \vee x_4)$

（3）构造有向图。有2n个顶点，每个顶点对应一个变量或变量的取反。有2m条边，每个子句 $\alpha \vee \beta$ 对应一条边 α 的取反指向 β 以及一条边 β 的取反指向 α 。

按要求建图即可。

（4）证明：若存在变量x，G中有一个同时包含 x 和 $\overline{x}$ 的强连通片，那么不存在使所有子句都满足的赋值。

- 如果有同时包含 x 和 $\overline{x}$ ，则有路径 $x \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow \overline{x}$ 和 $\overline{x} \rightarrow u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow x$
- 若 x 取TRUE，则 v_1 要取TRUE， v_2 也要取TRUE…… $\overline{x}$ 也要取TRUE，矛盾。
- 若 x 取FALSE，则 $\overline{x}$ 为TRUE， u_1 要取TRUE， u_2 也要取TRUE…… x 也要去TRUE，矛盾。

（5）证明（4）的逆命题。

要证明实例 I 是可满足的，则需要找到一种对每个变量真假状态的设置方案，见（6）

（6）

算法：

- 图 G 缩点得到图 G'
- 求图 G' 的拓扑序。
- 图 G 中每个顶点 x 的序号即为在 G' 中强连通片的拓扑序。
- 对每个变量 x 和其反值 $\overline{x}$ 的拓扑序：
    - $x < \overline{x}$ ，即可能 $x \to \overline{x}$ ， x 取FALSE。
    - $x = \overline{x}$ ，无解。
    - $x > \overline{x}$ ，即可能 $\overline{x} \to x$ ， x 取TRUE。
- 复杂度 $O(n+m)$

证明：

要证明算法正确性，则需要证明算法结束后，每个变量都已经确定了取值。

因为每个变量的真假都有确定的拓扑序，所以一定能按照这个规则确定真假值，并且没有冲突。

9.1

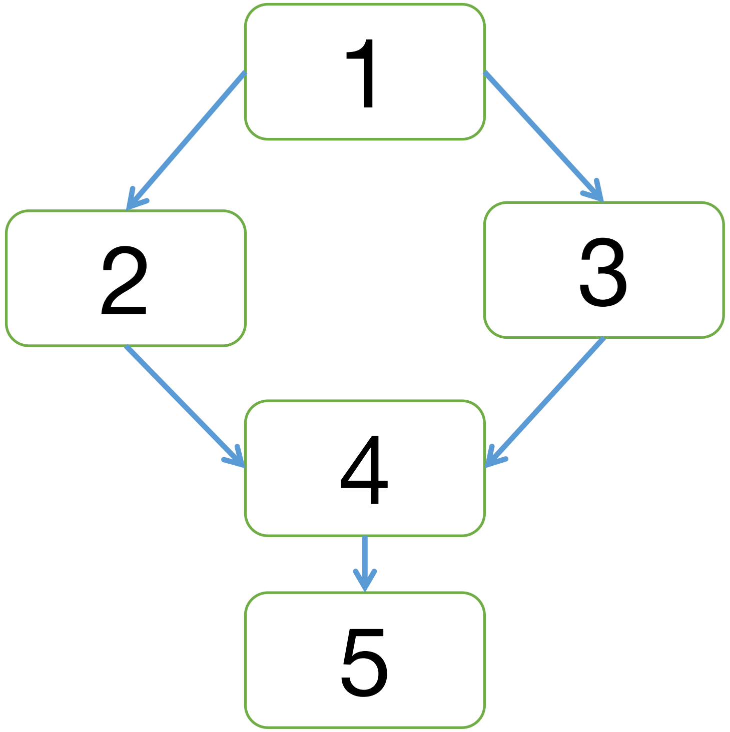
证明无向图BFS中不存在BE和DE。

- 假设在遍历 v 时发现了BE，指向了 w ，则在之前BFS中，队列弹出 w ，会发现 v 为 w 的为访问过的邻居，则BE其实是TE，矛盾。
- 假设在遍历 v 时发现了DE，指向了 w ，实际上还是TE，矛盾。

9.2

白色路径定理是否对BFS成立?

不成立。



刚发现3时，有3->4->5白色路径，但是5的祖先可能是2。

9.3

DFS可以判断是否为二分图。

寻找二分图等价于寻找长度为奇数的环。

- DFS不需要队列的空间开销，更适合深度大、节点度数小的图。
- BFS不需要进行递归，更适合深度小，节点度数大的图。

9.4

用DFS和BFS检测是否存在环。

- DFS
  - 有向图和无向图都是找BE（灰色节点）
- BFS
  - 有向图，找BE（黑色祖先），注意黑色节点不一定是BE，也可能是CE，需要额外判断是否是有共同parent。
  - 无向图，遇到灰色节点（CE）。

9.5

能否从无向连通图G中移除一条边，使G仍然连通。

本质还是找环，算法同9.4。

如何控制到 $O(n)$ ？遍历边的次数其实还是 $O(n)$ 而不是 $O(m)$ 。

（DFS无向图至多遍历n个点，检测到环后立刻返回）

9.8

求最小生成树。

（1）边权值均为1。

深度优先遍历树或者宽度优先遍历树就是最小生成树。

（2）m=n+10。

- 先随便遍历一次生成一个遍历树。
- 对剩下的11条边的每条边 uv ，从树上找 u 到 v 的路径，找到路径上最大边并与 uv 比较，如果 uv 更小则替换掉。

（3）边权值均为1或2。

- 先只考虑权重为1的边，DFS或者BFS遍历生成一个森林。
- 对森林的每棵树看作一个点（缩点），只考虑权重为2的边，遍历生成一个遍历树，即为所求。

分析：最小生成树边数是确定的，我们贪心的尽可能多地选取权为1的边。

9.10

求最短路径数量。

1）用BFS生成新图，只保留最短路径的边。

- 在bfs时记录每个点u到起始点d的距离d(u)。
- 对于bfs中TE边，直接加入G'。
- 对于bfs中的CE边(u,v)，若d(u)+1=d(v)，加入G'。
- V' = V。

2) 求c[u]。

- 对于点u，s到u的最短路径数c(u)满足以下关系：

  $c(u) = \sum c(i)$ ，其中i满足d(i)=d(u)-1且存在i到u的边。

- 动态规划。从s开始bfs，通过上一层的c(i)计算下一层的c(j)，直到c(u)。
- 需要在G'上进行BFS，以去除冗余的边。