割边和边双连通分量的求解

主讲人：邓哲也

朴素做法

联系割边的定义，依次枚举每条边，删除它，然后遍历整个图，得到图的连通分量个数，如果大于等于 2，则这条边就是割边。

时间复杂度：$O(n^3)$

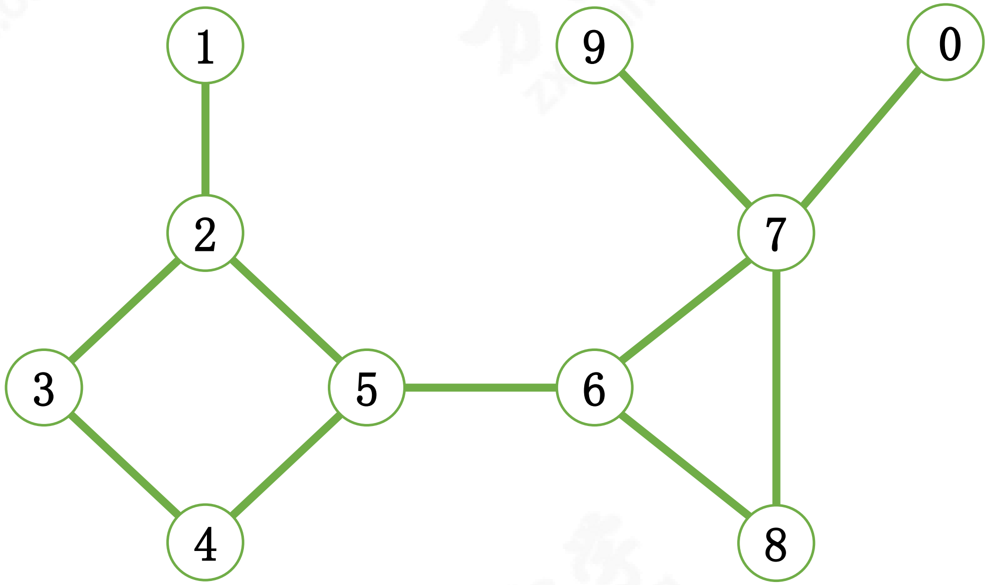
Tarjan算法求割边

与之前求割点的 Tarjan 算法类似，我们只要稍作改动，就可以同样通过一次遍历，求出图中所有的割边。

时间复杂度：$O(n + m)$

Tarjan算法求割边

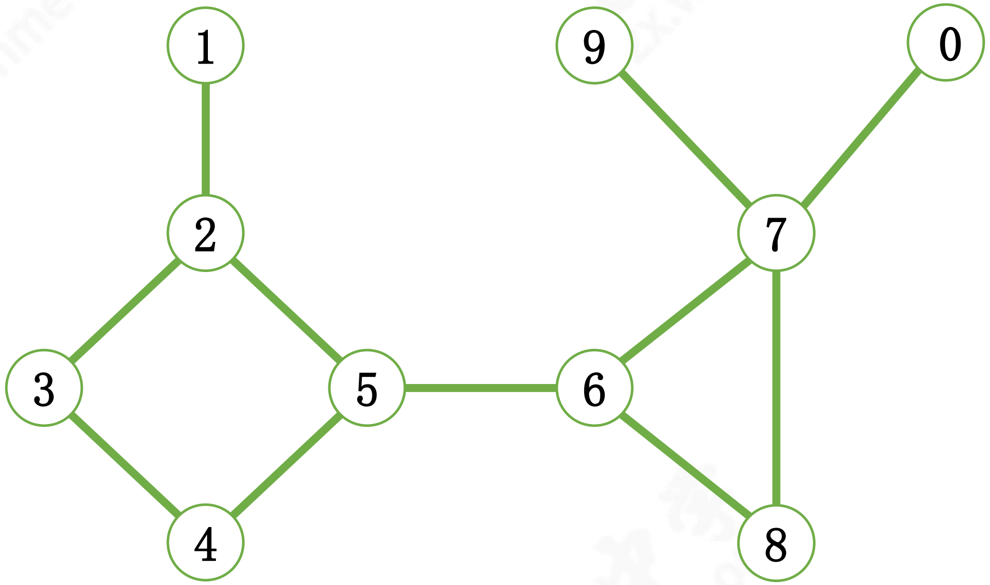
以这张图为例，我们来看看 Tarjan 算法是如何运行的。

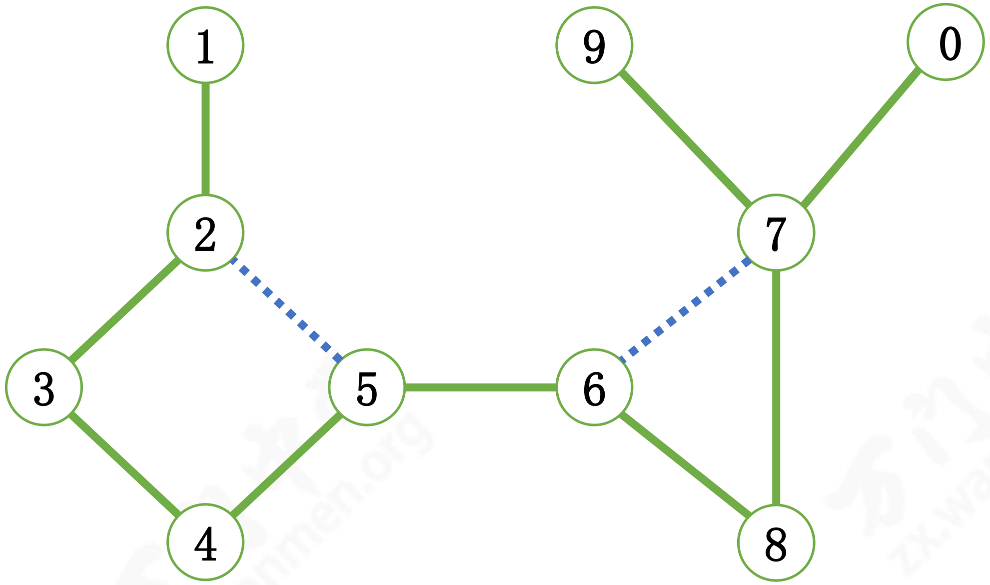


Tarjan算法求割边

首先，任选一个点进行 DFS。

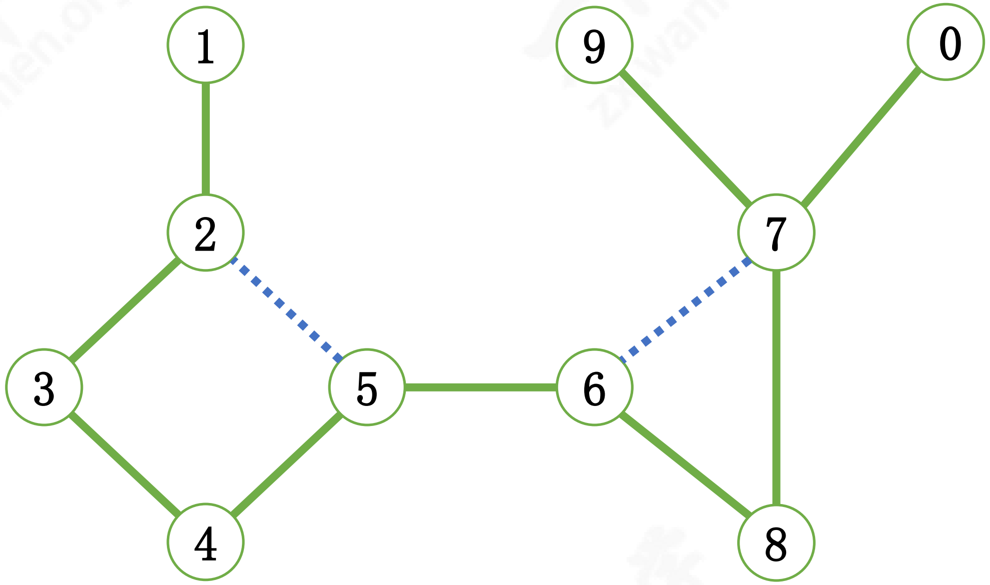
比如从 5 号点开始 DFS，按顺序经过：5 4 3 2 1 6 8 7 0 9

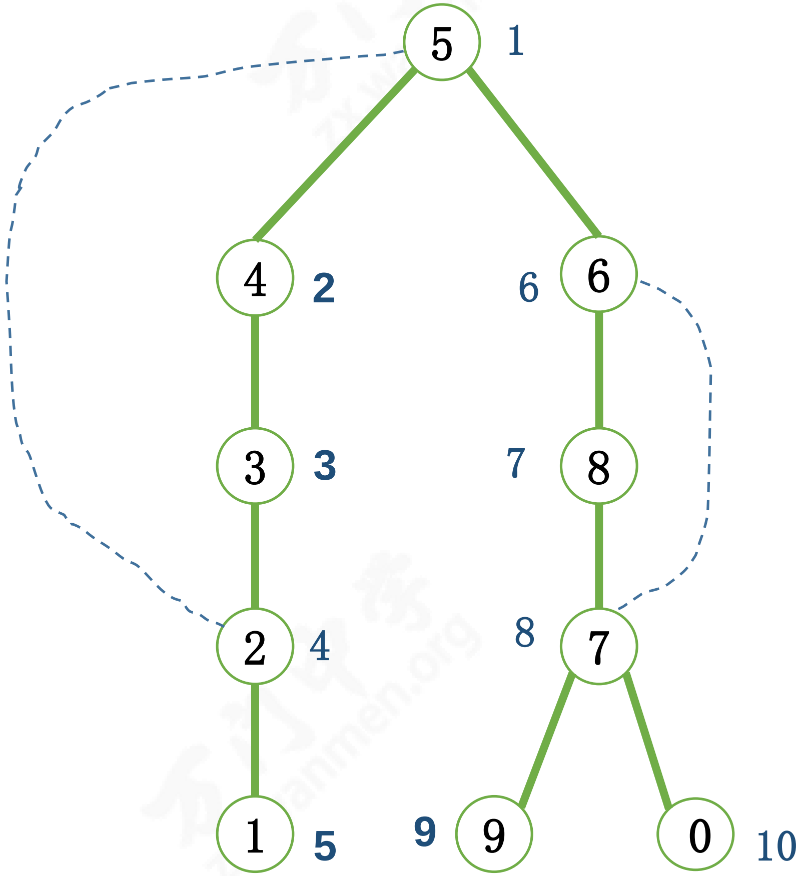




Tarjan算法求割边

根为 5 的深度优先搜索生成树。

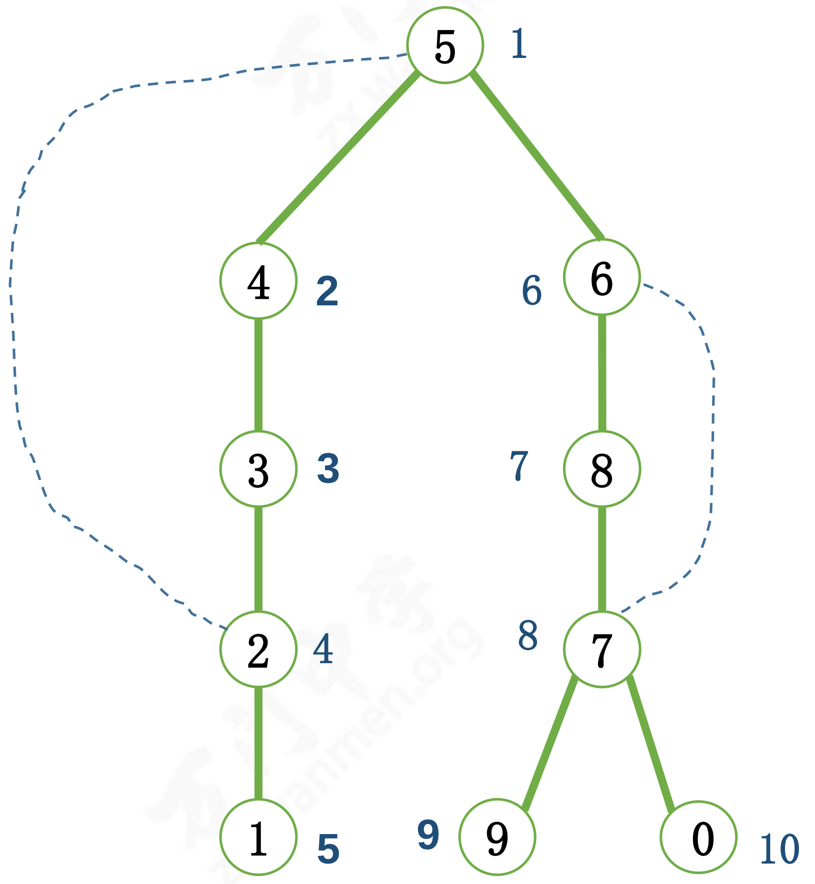




Tarjan算法求割边

dfn[] 表示 dfs 时访问的顺序。

显然，在深度优先搜索生成树中，如果 u 是 v 的祖先，则一定存在 dfn[u] < dfn[v]，表示 u 比 v 先被访问到。



Tarjan算法求割边

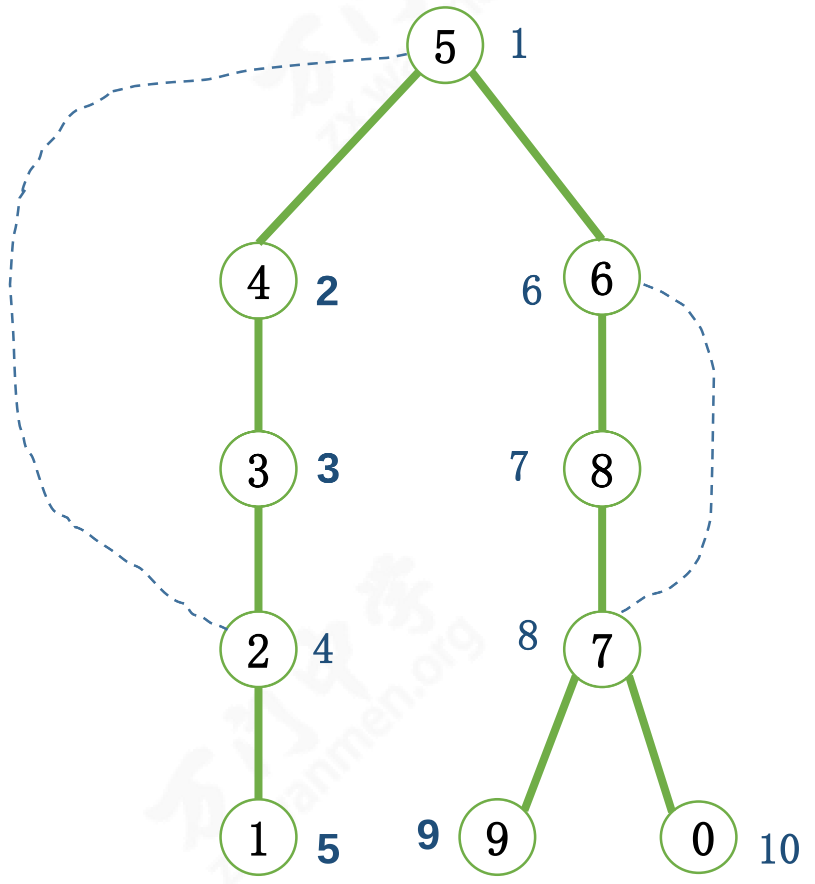
对 G 的每个顶点 u 定义一个 low 值，

low[u] 表示从 u 或 u 的子孙出发通过返向边可以到达的最低深度优先数(dfn)。

low[u] = min{ dfn[u],

min { low[v] | v 是 u 的一个孩子 },

min { dfn[v] | (u, v)是一条返向边 } }.



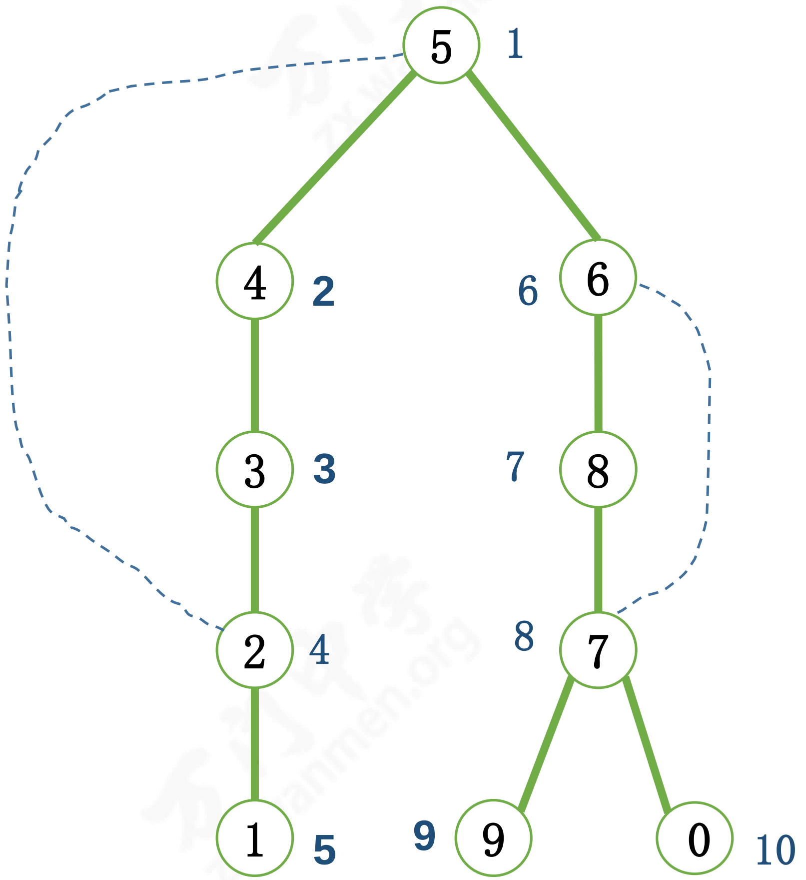
Tarjan算法求割点回顾

顶点 u 是割点的充要条件：

（1）如果 u 是深度优先搜索生成树的根，则 u 至少有两个子女。

（2）如果 u 不是生成树的根，则它至少有一个孩子 v，从 v 出发不可能通过 v、v的子孙，以及一条回边到达 u 的祖先。

u 要么是具有两个以上子女的深度优先生成树的根，要么存在一个 u 的子节点 v 使得 low[v] >= dfn[u].



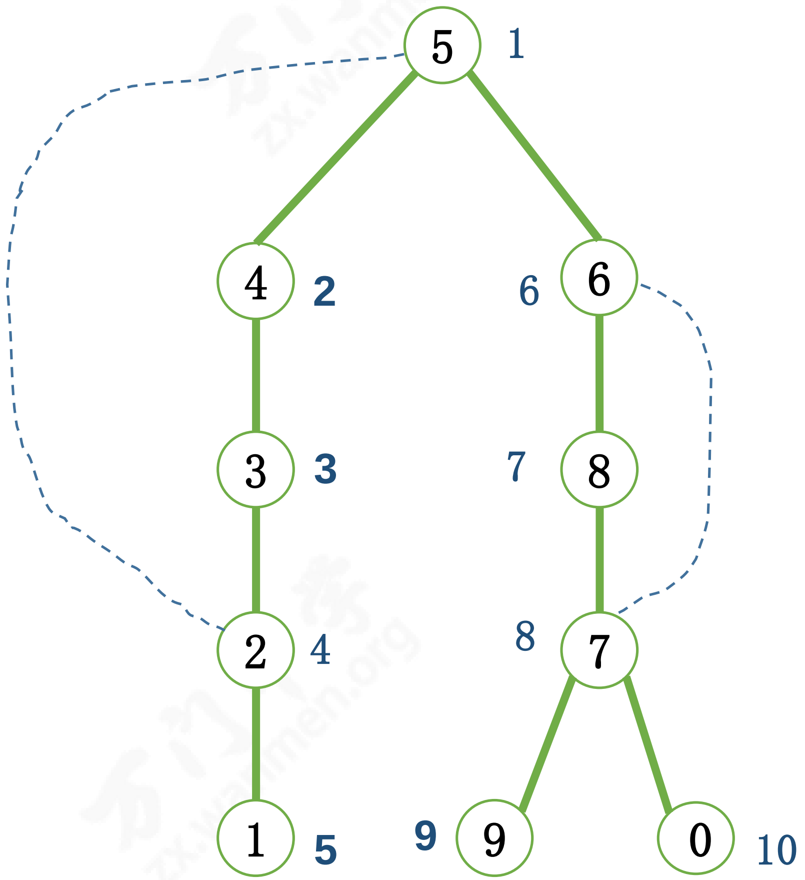
Tarjan算法求割边

(u, v)是割边的充要条件：

（1）(u, v) 是生成树边

（2）对于 u 的孩子 v，从 v 出发不可能通过 v、v 的子孙，以及一条回边到达 u 或 u 的祖先。

(u, v) 为生成树中的边，且 dfn[u] < low[v].



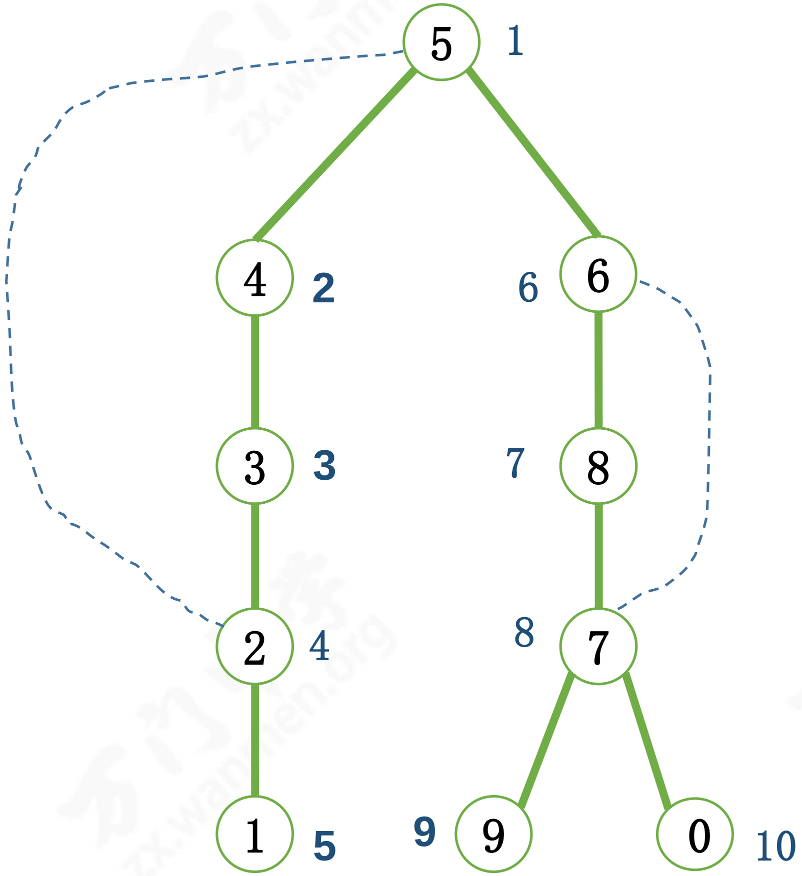
Tarjan算法求割点和割边的比较

割点：

u 要么是具有两个以上子女的深度优先生成树的根，要么存在一个 u 的子节点 v 使得 low[v] >= dfn[u].

割边：

(u，v) 为生成树中的边，且 dfn[u] < low[v].

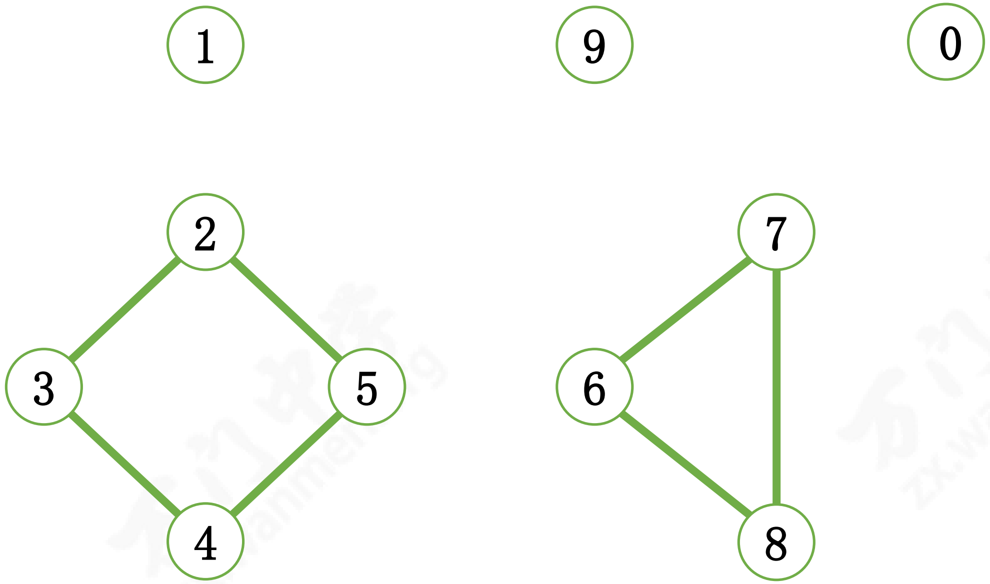


边双连通分量的求解

与点双连通分量的求解相比，边双连通分量的求法十分简单。

只要在求出所有桥之后，把桥删除，原图就变成了多个连通块，每个连通块就是一个边双连通分量。

桥不属于任何一个边双连通分量，其余的边和顶点都属于且只属于一个边双连通分量。



下节课再见