割点的求解

主讲人：邓哲也

朴素做法

联系割点的定义，依次枚举每个顶点，删除它和与它相关联的边，然后遍历整个图，得到图的连通分量个数，如果大于等于 2，则该顶点就是割点。

时间复杂度：$O(n^3)$

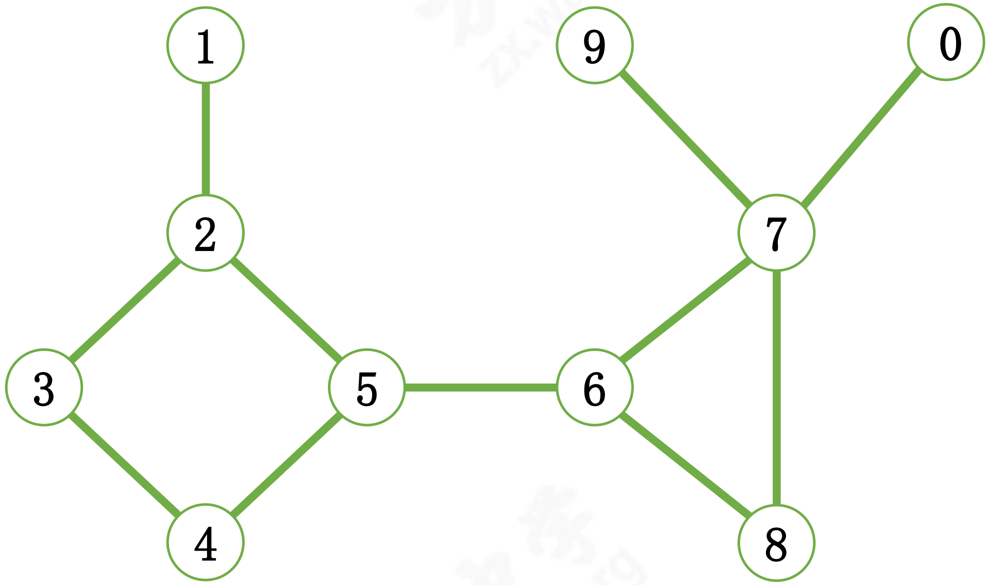
Tarjan算法求割点

这里我们介绍一种只需从某个顶点出发进行一次遍历，就可以求出图中所有割点的 Tarjan 算法。

时间复杂度：O(n + m)

Tarjan算法求割点

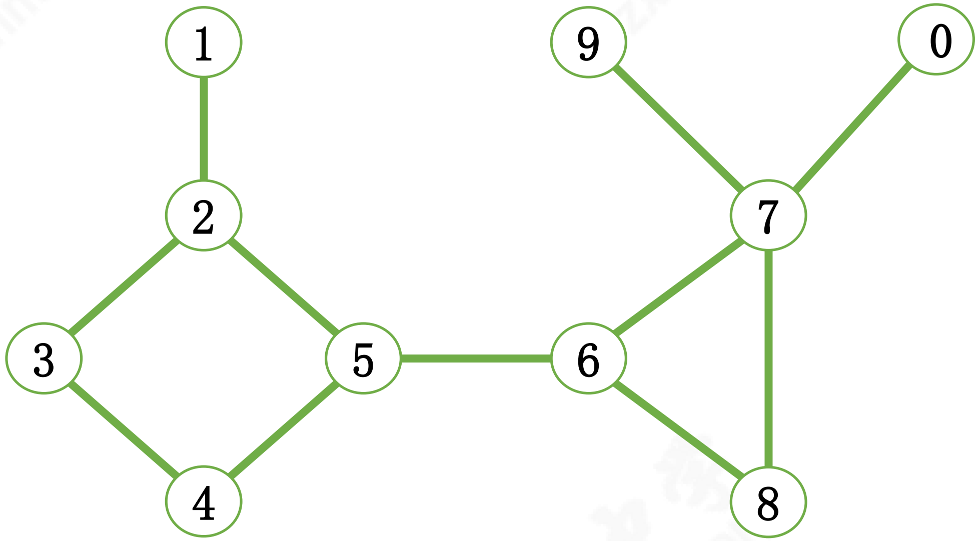
以这张图为例，我们来看看 Tarjan 算法是如何运行的。

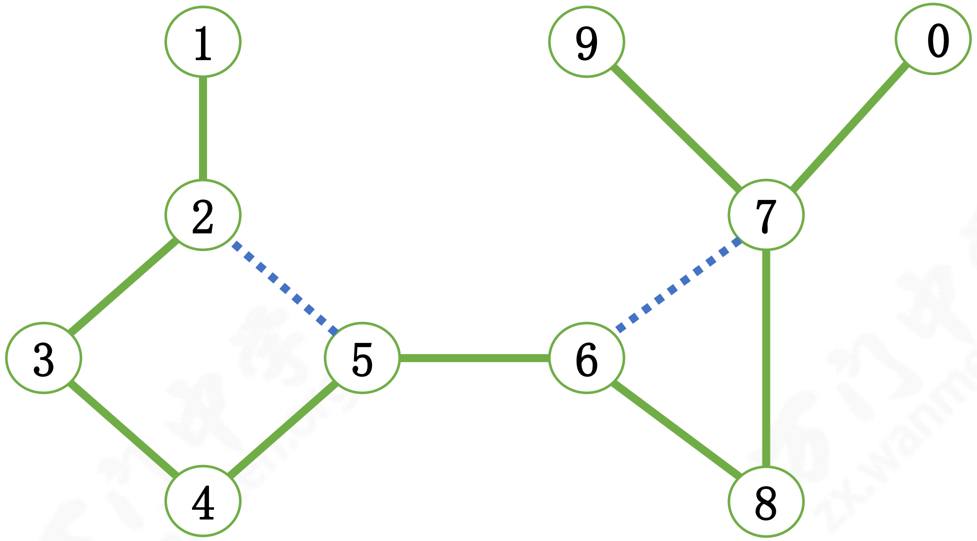


Tarjan算法求割点

首先，任选一个点进行 DFS。

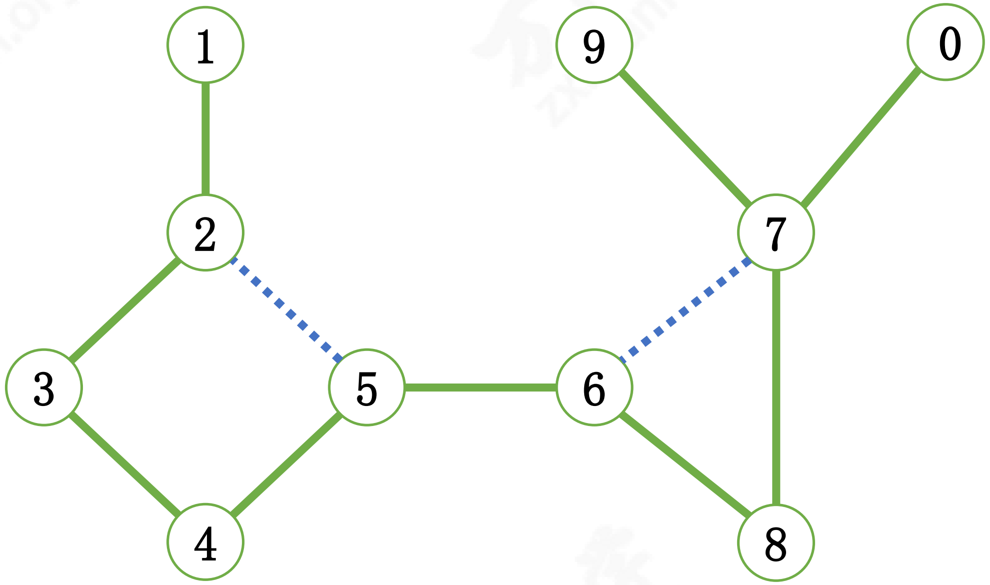
比如从 5 号点开始 DFS，按顺序经过：5 4 3 2 1 6 8 7 0 9

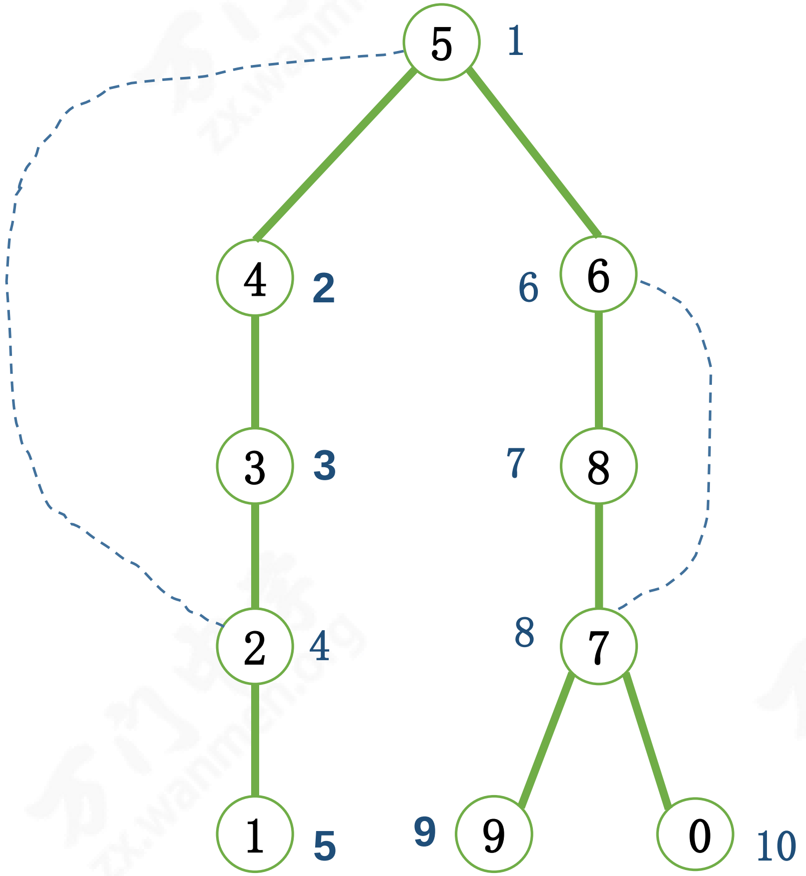




Tarjan算法求割点

根为 5 的深度优先搜索生成树。

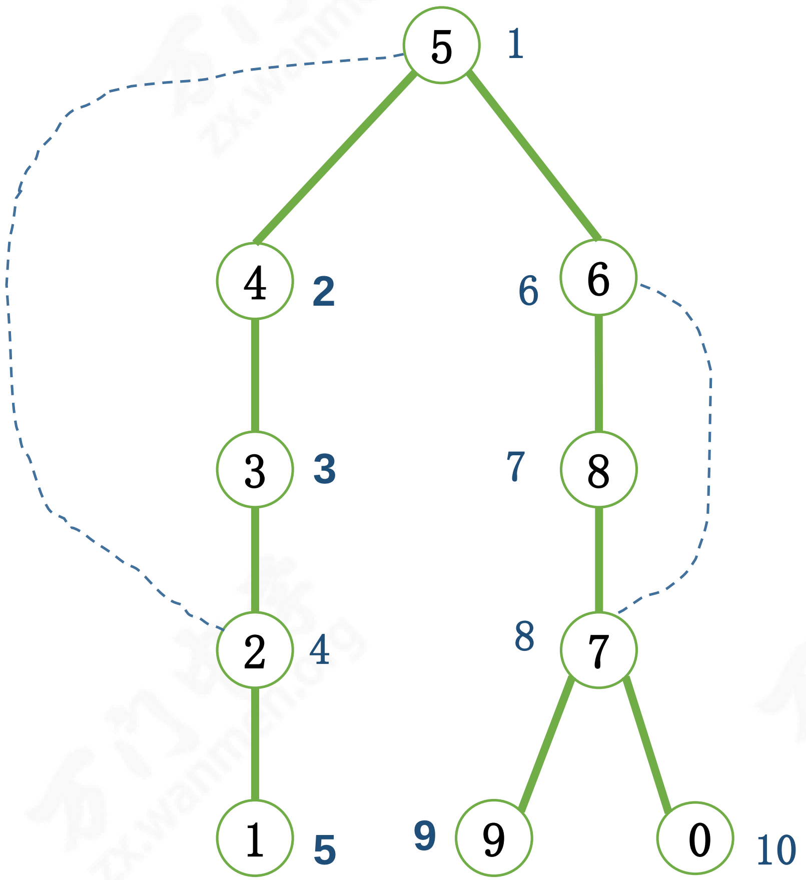




Tarjan算法求割点

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。



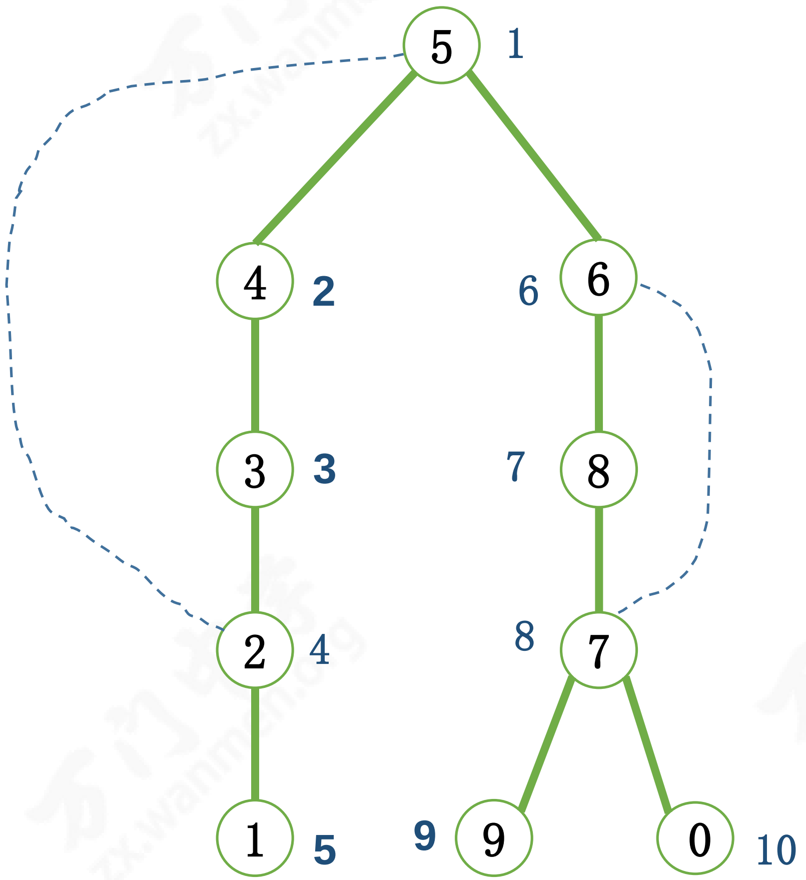
Tarjan算法求割点

重新回到原来的图 G。

G 中的边可以分为两种：

1. 生成树的边。例如：(4, 3), (7, 9)
2. 返向边。例如图中虚线表示的边(2, 5), (6, 7)

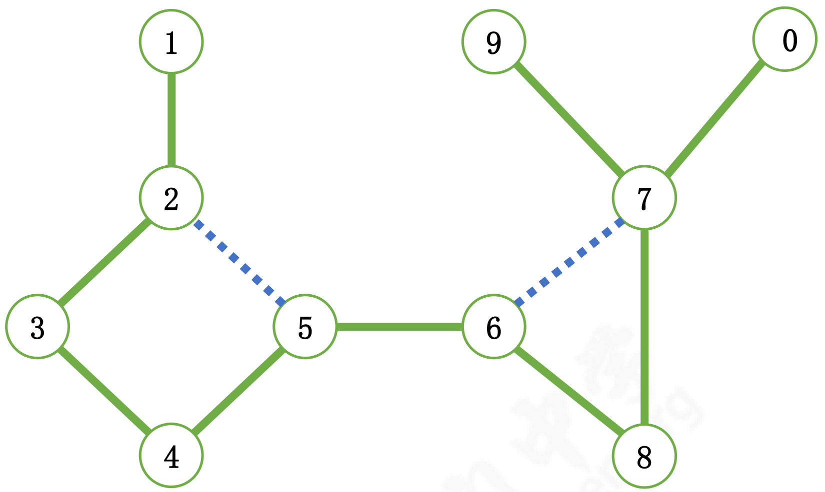
由 DFS 的性质可得，不存在除了上述两种边之外的边。

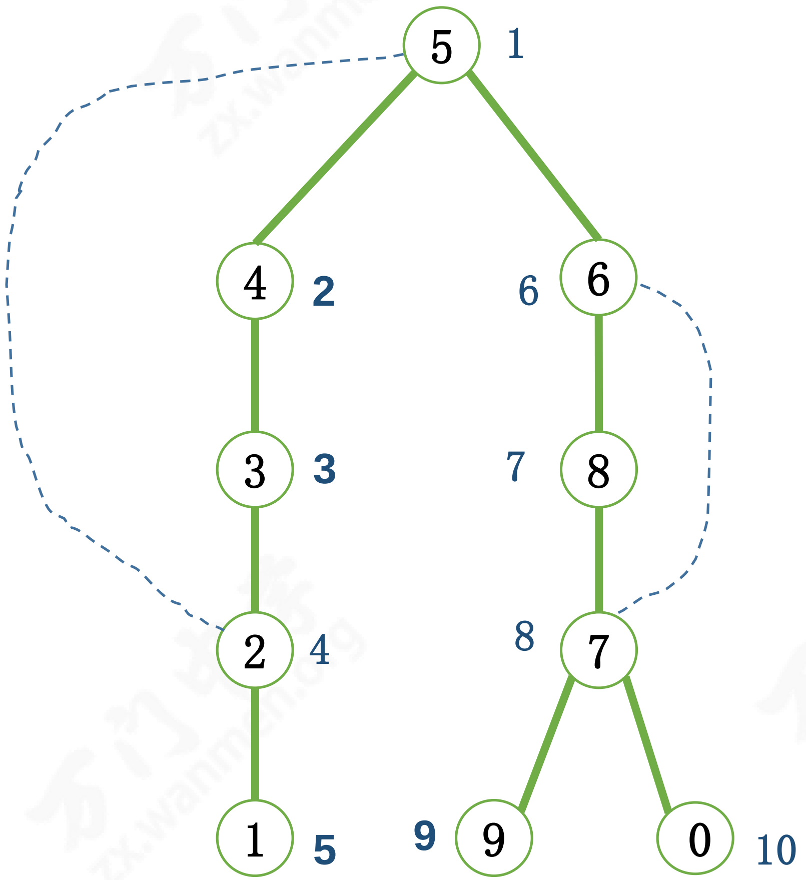


Tarjan算法求割点

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。

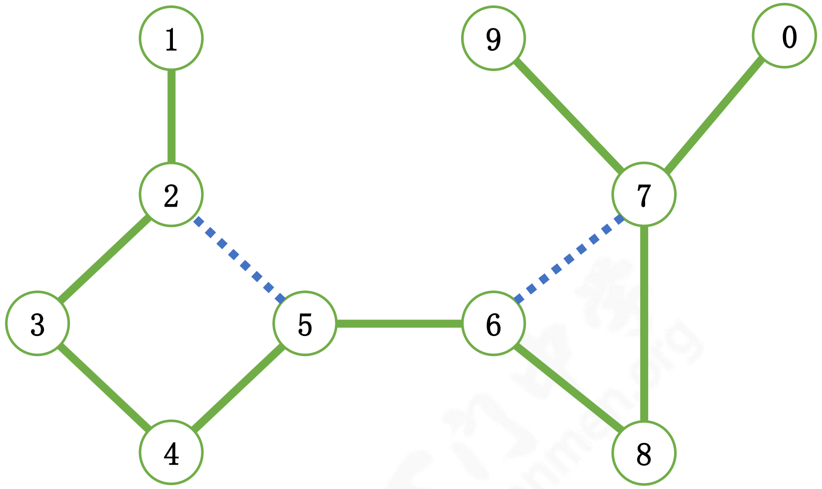


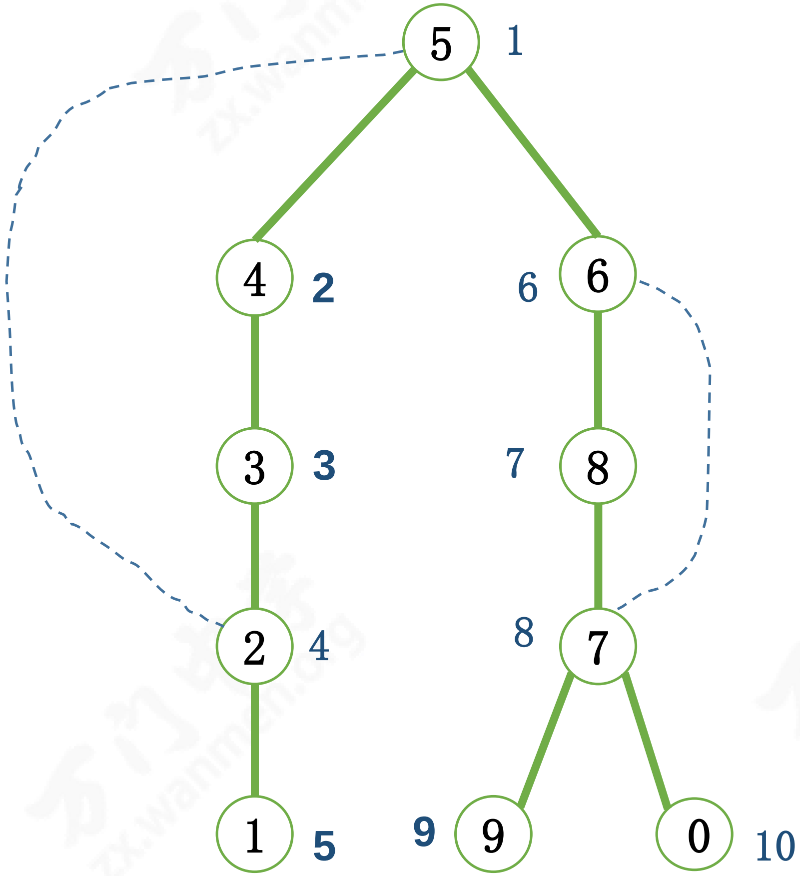


Tarjan算法求割点

顶点 u 是割点的充要条件（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。





Tarjan算法求割点

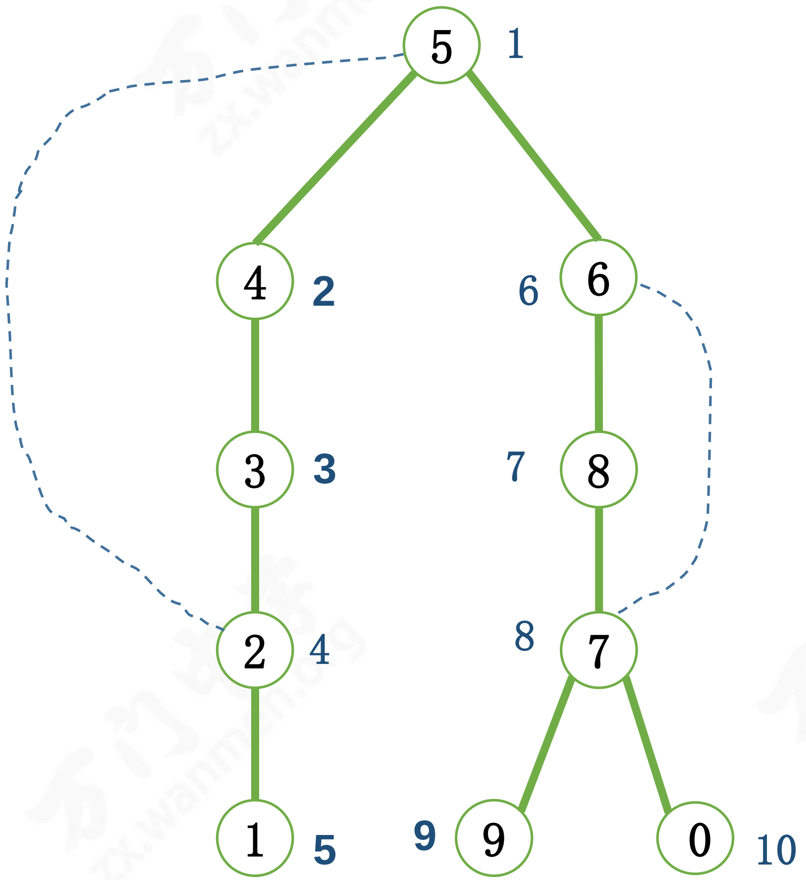
对 G 的每个顶点 u 定义一个 low 值，

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

dfn[u]

顶点 u 本身的深度优先数

min { low[v] | v 是 u 的一个孩子 }

它的孩子顶点 v 的 low[v] 的最小值

min { dfn[v] | (u, v)是一条返向边 }

它直接通过返向边可以到达的最低优先数。

Tarjan算法求割点

综上所述，顶点 u 是割点的充要条件是：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].

因此只需要进行一次 DFS，求出 dfn[] 和 low[]，就可以一次性求出图中的全部割点。

下节课再见