

07 图应用 | 07A1 双连通分量：判定准则

#数据结构邓神

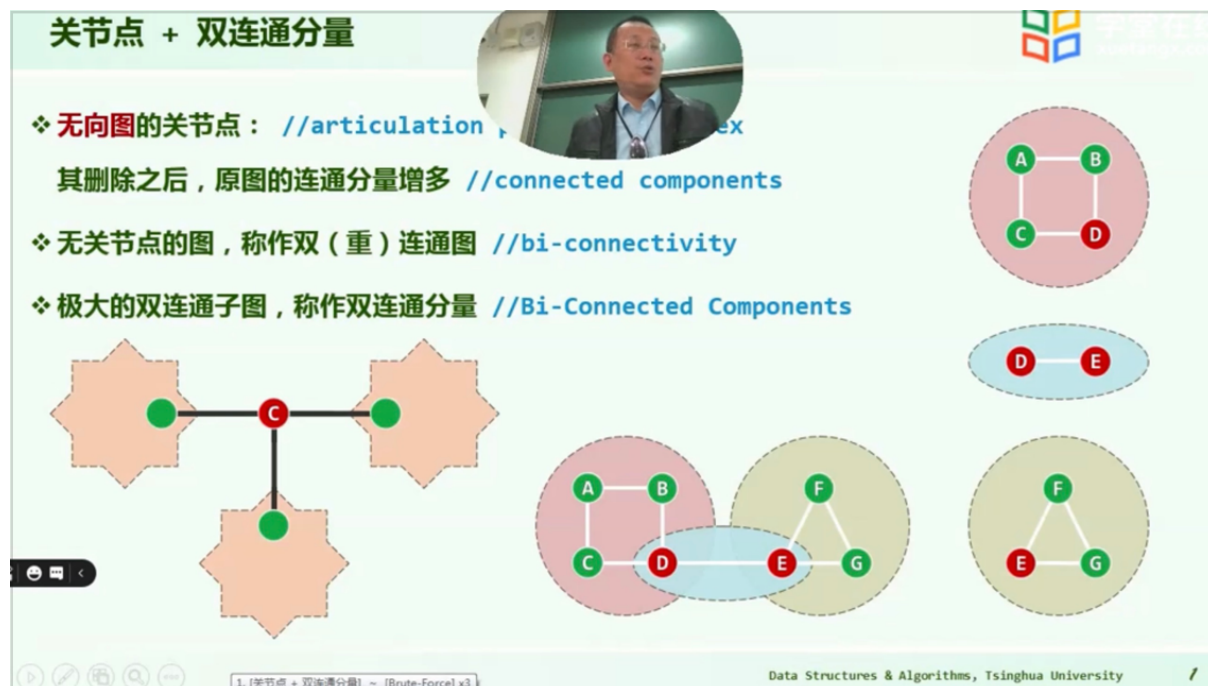
连通：两点之间无论是有向还是无向只要有就是连通的

关节点 + 双连通分量

❖ 无向图的关节点：//articulation point
其删除之后，原图的连通分量增多 //connected components

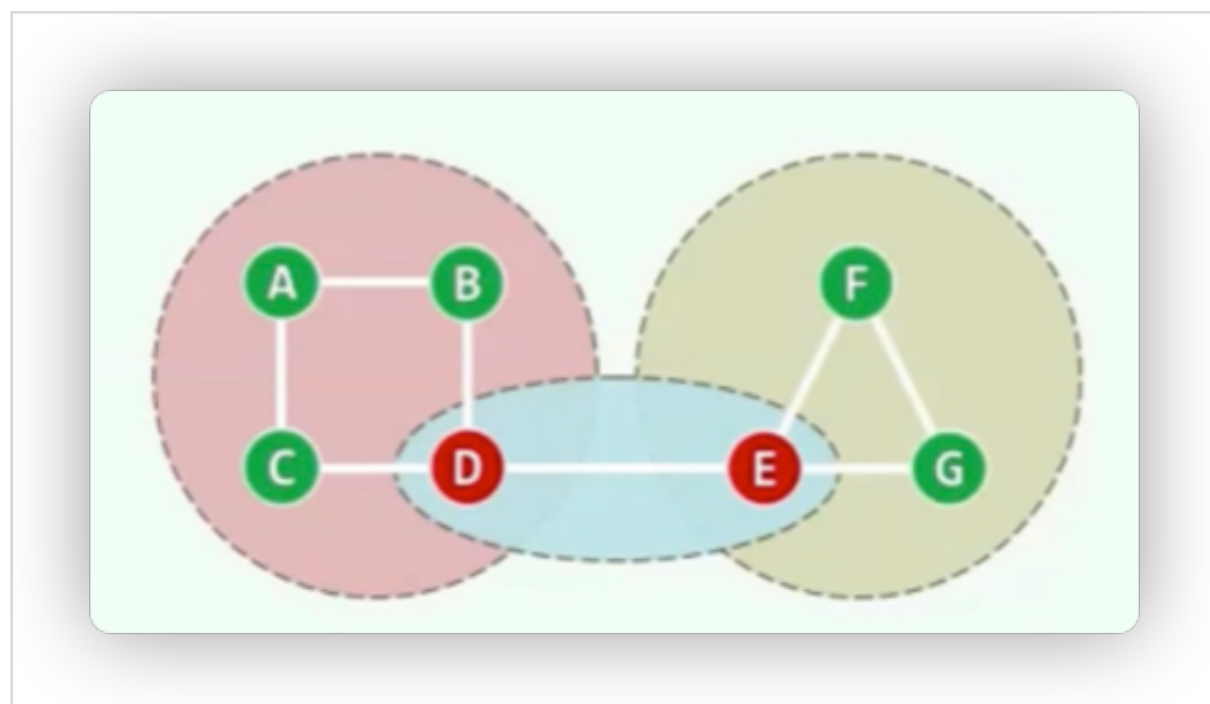
❖ 无关节点的图，称作双（重）连通图 //bi-connectivity

❖ 极大的双连通子图，称作双连通分量 //Bi-Connected Components

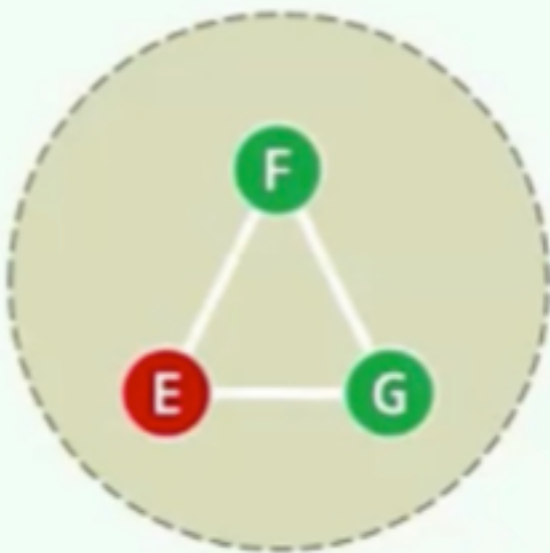
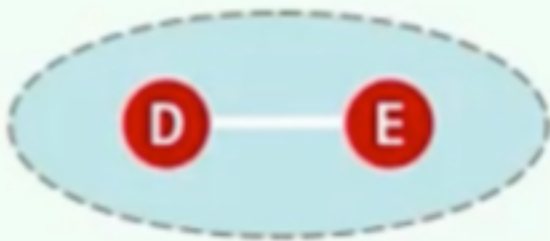
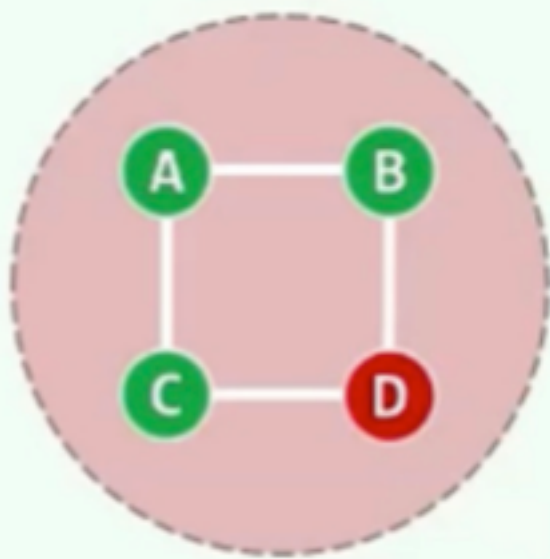


关节点（关键点）：红色让两个连通域的连在一起的重要节点

每一个不存在关节点的子图叫做双连通分量



存在三个双连通分量

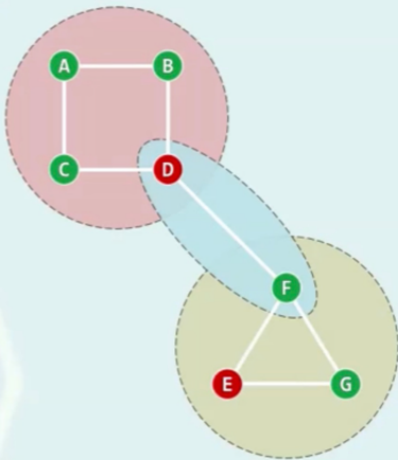


我们如何找到双连通分量呢？

Brute-Force 蛮力 平凡的 没有任何效率的

Brute-Force

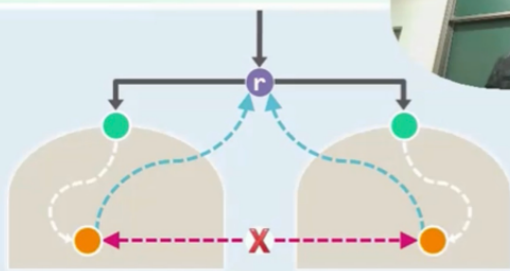
- ❖ 给定无向图，如何确定各BCC？
- ❖ 先考察简单的版本：如何确定关节点？
- ❖ 蛮力：对每一顶点 v ，通过遍历检查 $G \setminus \{v\}$ 是否连通
- ❖ 共需 $O(n * (n + e))$ 时间，太慢！
- 而且，即便找出关节点，各BCC仍需确定
- ❖ 改进：从任一顶点出发，构造DFS树
- 根据DFS留下的标记，甄别是否关节点
- ❖ 比如，叶节点绝不可能是关节点 //为什么？



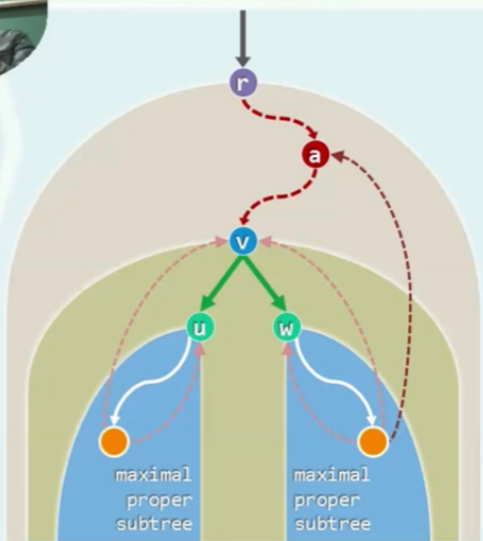
把每一个点都摘掉，考察连通域都数量
但是复杂度太高了！

叶子一定不是关键点

非叶节点



- ❖ 根 r ：必须至少有2棵子树
- ❖ 内部节点 v ：
 - 有某个孩子 u ，而 $subtree(u)$ 不能经由BACKWARD边，联接到 v 的任何真祖先 a
- ❖ 此时， $\{v\} = BCC(u) \cap BCC(\text{parent}(v))$



经过DFS算法后一定会获得一颗DFS树

我们首先来考虑叶子节点是不是关节点

删除叶子节点不会使得连通区域增加,或者说叶子节点没有子树

树根：

必须要有两个子树 只要度 ≥ 2 就一定是！！！！

因为DFS算法必然会扫描完成一个连通域，也就表示他们中间必然没有连通的地方

非叶节点

- ❖ 根 r : 必须至少有2棵子树
- ❖ 内部节点 v :
 - 有某个孩子 u , 而 $\text{subtree}(u)$ 不能经由BACKWARD边, 联接到 v 的任何真祖先 a
- ❖ 此时, $\{v\} = \text{BCC}(u) \cap \text{BCC}(\text{parent}(v))$

非叶子节点（内部节点）

Highest Connected Ancestor

- ❖ $\text{hca}(v) = \text{subtree}(v)$ 经后向边能抵达
- ❖ 由括号引理: dTime 越小的祖先, 辈份越高
- ❖ DFS过程中, 一旦发现后向边 (v, u)
 - 即取: $\text{hca}(v) = \min(\text{hca}(v), \text{dTime}(u))$
- ❖ DFS(u)完成并返回 v 时
 - 若有: $\text{hca}(u) < \text{dTime}(v)$
 - 即取: $\text{hca}(v) = \min(\text{hca}(v), \text{hca}(u))$
- ❖ 否则, 即可断定: v 系关节点, 且 $\{v\} + \text{subtree}(u)$ 即为一个BCC

我们需要在向下DFS探索中不断看BackEdge判断他所指向的是否比当前的 v 更高, 那就不是, 如果没有那就是