6.图

（c）广度优先搜索

邓俊辉

deng@tsinghua.edu.cn

算法

- ❖ 始自顶点s的广度优先搜索 Breadth-First Search

  访问顶点s

  依次访问s所有 尚未访问 的邻接顶点

  依次访问它们 尚未访问 的邻接顶点

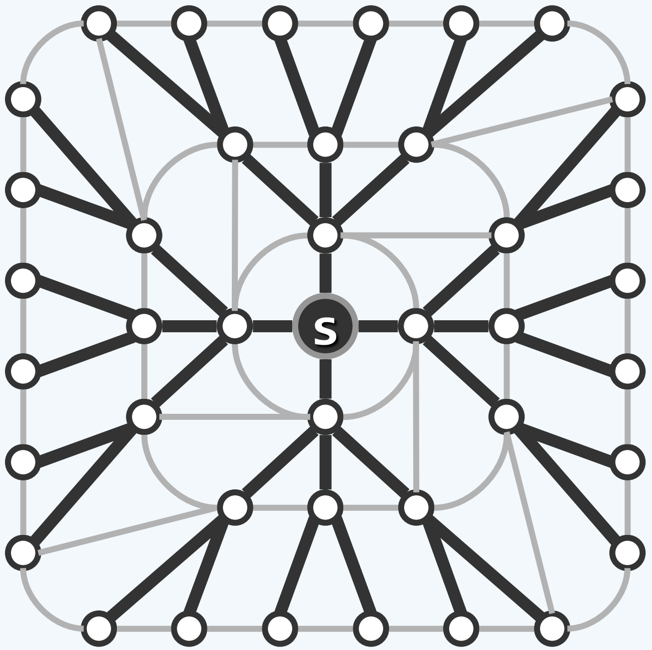
  . . .

  如此反复

  直至没有 尚未访问 的邻接顶点

- ❖ 以上策略及过程完全等同于 树 的 广度优先遍历

- ❖ 事实上，BFS也的确会构造出原图的一棵 支撑树（BFS tree）



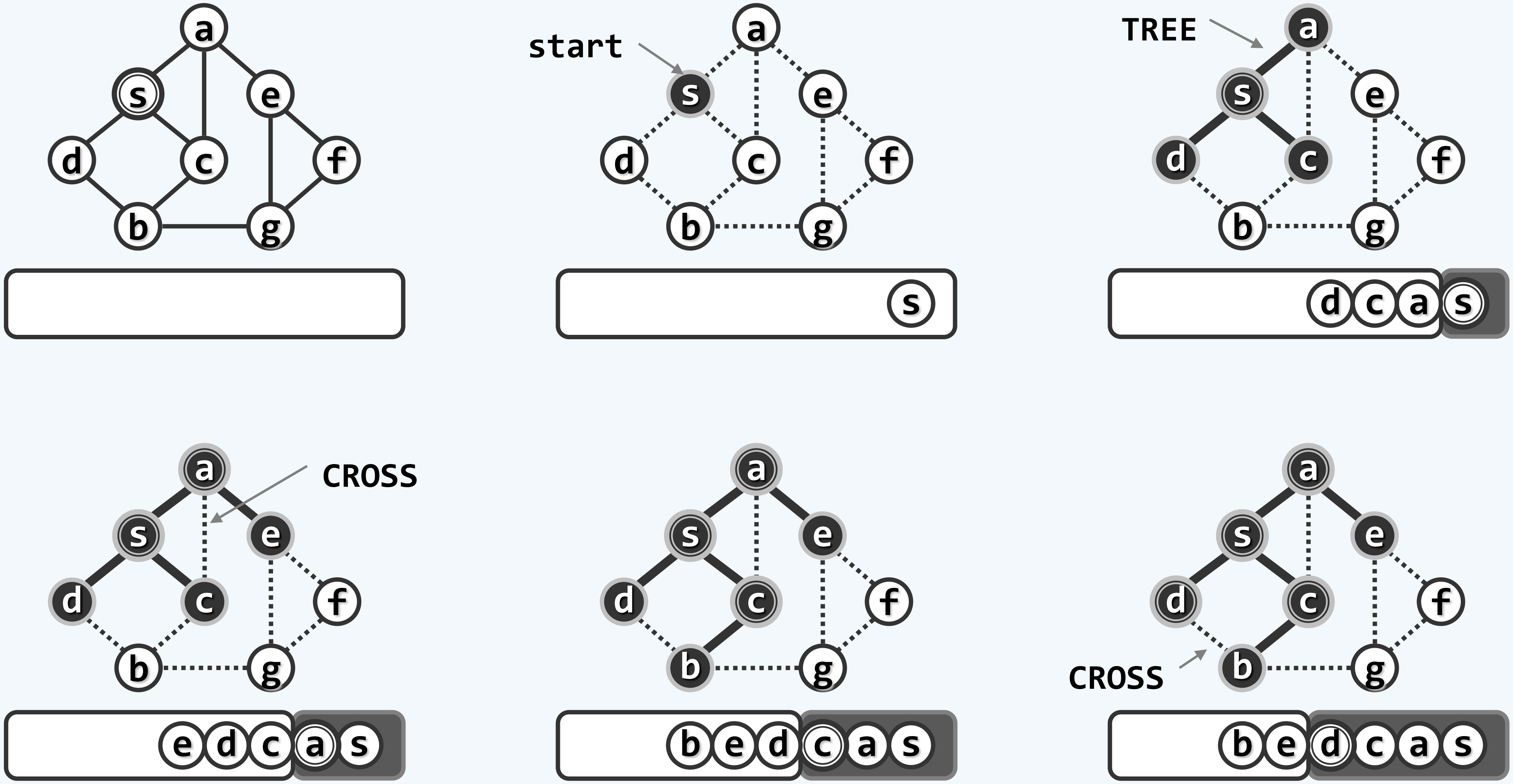
Graph::BFS()

```
❖ template <typename Tv, typename Te> //顶点类型、边类型
  void Graph<Tv, Te>::BFS( int v, int & clock ) {
     Queue<int> Q; status(v) = DISCOVERED; Q.enqueue(v); //初始化
     while ( !Q.empty() ) { //反复地
        int v = Q.dequeue();
        dTime(v) = ++clock; //取出队首顶点v，并
        for ( int u = firstNbr(v); -1 < u; u = nextNbr(v, u) ) //考察v的每一邻居u
           /* ... 视u的状态，分别处理 ... */
        status(v) = VISITED; //至此，当前顶点访问完毕
     }
  }
```

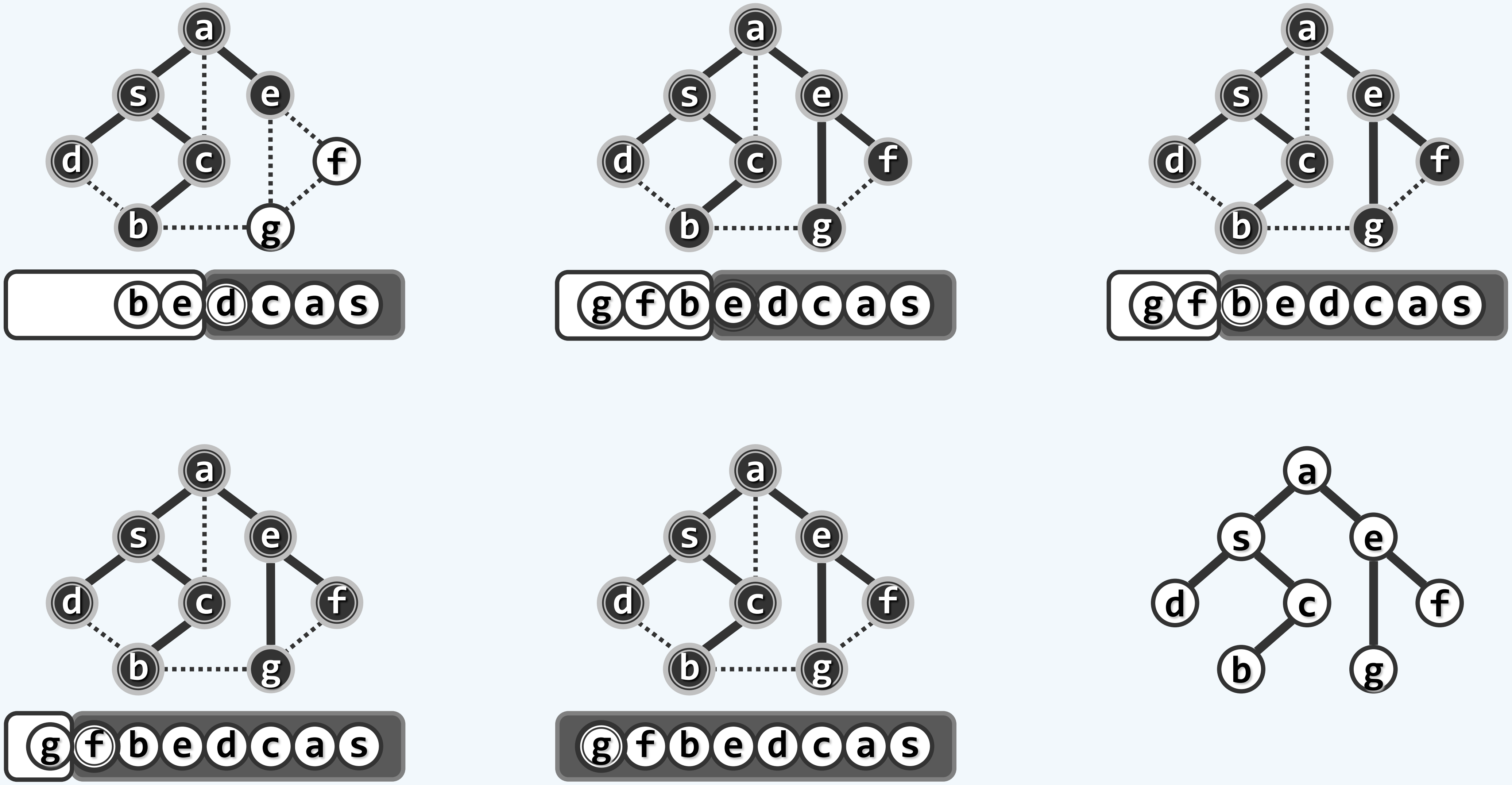
Graph::BFS()

```
while ( !Q.empty() ) { //反复地
    int v = Q.dequeue(); dTime(v) = ++clock; //取出队首顶点v，并
    for ( int u = firstNbr(v); -1 < u; u = nextNbr(v, u) ) //考察v的每一邻居u
        if ( UNDISCOVERED == status(u) ) { //若u尚未被发现，则
            status(u) = DISCOVERED; Q.enqueue(u); //发现该顶点
            type(v, u) = TREE; parent(u) = v; //引入树边
        } else //若u已被发现（正在队列中），或者甚至已访问完毕（已出队列），则
            type(v, u) = CROSS; //将(v, u)归类于跨边
    status(v) = VISITED; //至此，当前顶点访问完毕
}
```

实例（无向图）



实例（无向图）

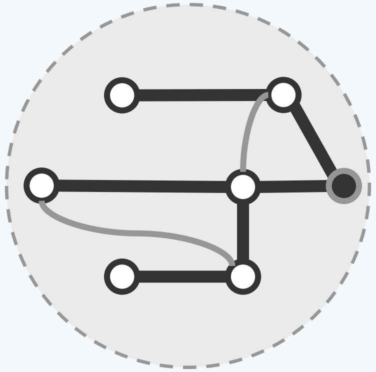


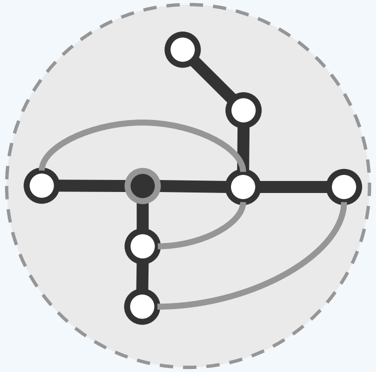
连通分量与可达分量

❖ 问题

给定 无向图 ，找出其中任一顶点s所在的 连通分量

给定 有向图 ，找出源自其中任一顶点s的 可达分量



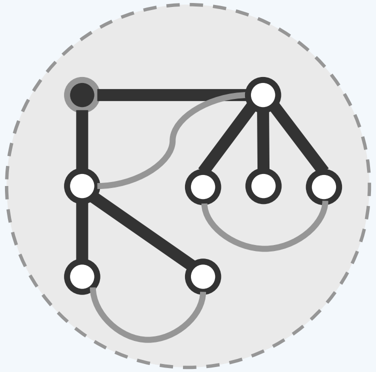


❖ 算法

从s出发做BFS

输出所有 被发现 的顶点

队列为空后立即终止，无需考虑其它顶点



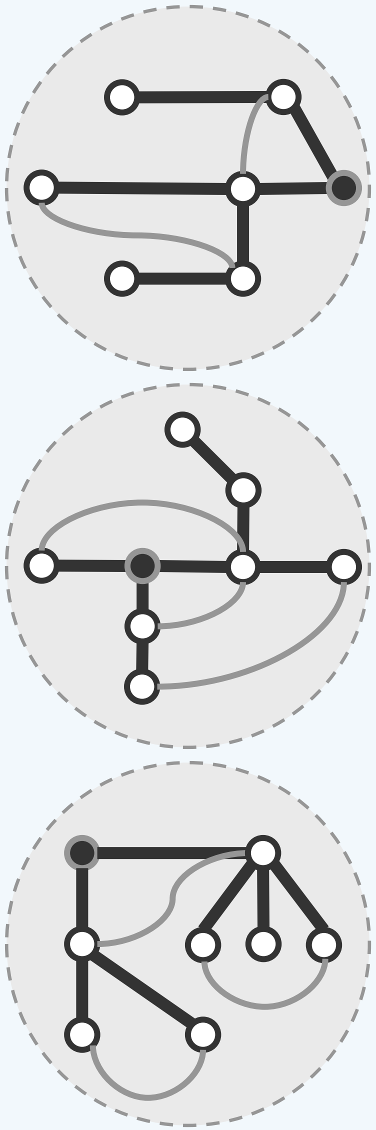
❖ 若图中包含 多个 连通/可达分量

又该如何保证对 全图 的遍历呢？

Graph::bfs()

```
❖ template <typename Tv, typename Te> //顶点类型、边类型
  void Graph<Tv, Te>::bfs( int s ) { //s为起始顶点
     reset(); int clock = 0; int v = s; //初始化Θ(n + e)
     do //逐一检查所有顶点，一旦遇到尚未发现的顶点
        if ( UNDISCOVERED == status(v) ) //累计Θ(n)
           BFS( v, clock ); //即从该顶点出发启动一次BFS
     while ( s != ( v = ( ++v % n ) ) );
        //按序号访问，故不漏不重
  } //无论共有多少连通/可达分量...
```

❖ bfs()均可遍历它们，而且自身累计仅需线性时间...



复杂度

- 考查无向图...
- bfs()的初始化（reset()） $\mathcal{O}(n + e)$
- BFS()的迭代 $\mathcal{O}(n + 2e)$

  外循环（while (!Q.empty())），每个顶点只进入1次，累计n次 $\mathcal{O}(n)$

  内循环（枚举v的每一邻居），每个邻居至多进入1次，累计deg(v)次

  采用邻接矩阵 $\boxed{\mathcal{O}(n)}$

  采用邻接表 $\boxed{\mathcal{O}(1 + deg(v))}$

  总共 $= \mathcal{O}(\ \sum_v (1 + deg(v))\ ) = \mathcal{O}(n + 2e)$

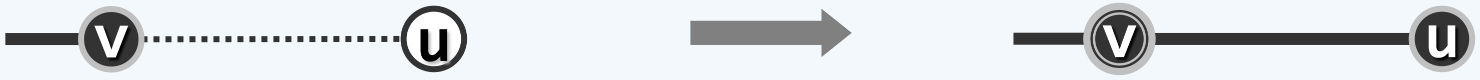
- 整个算法：$\mathcal{O}(n + e) + \mathcal{O}(n + 2e) = \boxed{\mathcal{O}(n + e)}$
- 有向图呢？亦是如此！

边分类

❖ 经BFS后，所有边将确定方向，且被分为两类 //有向图有何不同？

tree edges + cross edges

❖ 树边(v, u)



联边之前：v $\boxed{\text{DISCOVERED}}$ && u $\boxed{\text{UNDISCOVERED}}$

联边之后：v == parent(u)

❖ 跨边(v, u)

无向图：只可能v和u均为$\boxed{\text{DISCOVERED}}$

有向图：还可能v是$\boxed{\text{DISCOVERED}}$、u是$\boxed{\text{VISITED}}$

BFS树/森林

❖ 对于每一连通/可达分量，bfs()进入BFS(v)恰好1次（v为该分量的起始顶点）

❖ 进入BFS(v)时，队列为空

v所属分量内的每个顶点

迟早会以UNDISCOVERED状态进队1次

进队后随即转为DISCOVERED状态，并生成一条树边

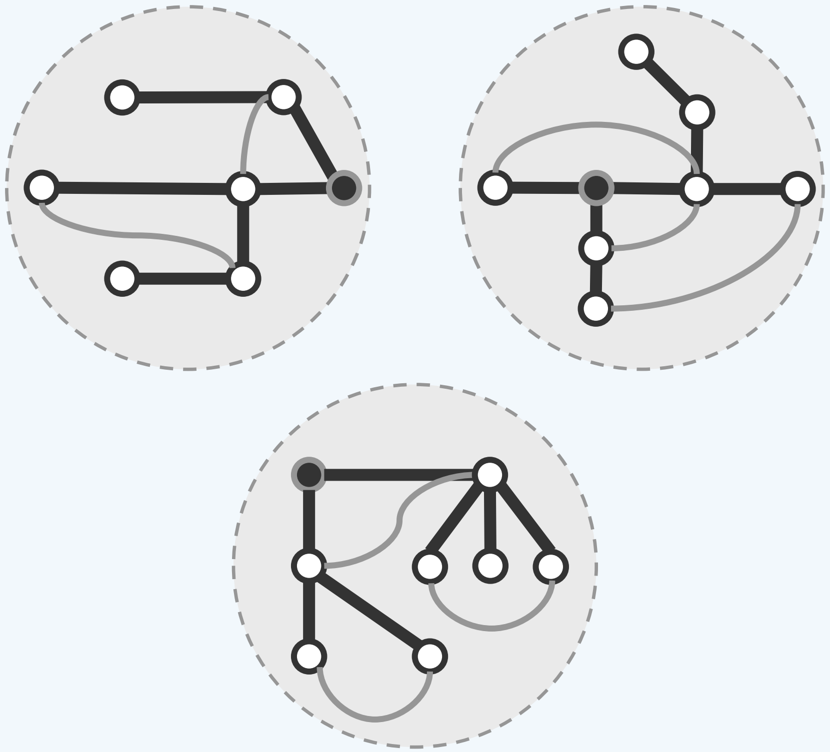
迟早会出队并转为VISITED状态

退出BFS(v)时，队列为空

❖ BFS(v)以v为根，生成一棵BFS树

❖ bfs()生成一个BFS森林包含

$\boxed{c}$棵树、$\boxed{n - c}$条树边和$\boxed{e - n + c}$条跨边



最短路径

- 从 $\boxed{\text{无向图}}$ 中顶点s出发，到任一顶点v的所有路径中

  长度最短者 $\pi(s, v) = ?$

  （最短）距离 $= dist(v) = |\pi(s, v)| = ?$

  退化情况：可能有多条 //任取其一

- 在BFS过程中的任一时刻

  队列中顶点按dist()单调排列

  队列中相邻顶点，dist()相差不超过1

  队首、队末顶点，dist()相差不超过1

  由树边联接的顶点，dist()恰好相差1

  由跨边联接的顶点，dist()至多相差1 //至少相差 $\boxed{2}$ 的情况呢？

- BFS树中从s到v的路径，即是 $\pi(s, v)$