06-D1

图

广度优先搜索：算法

邓俊辉

deng@tsinghua.edu.cn

Breadth-First Search

❖ 始自顶点s的广度优先搜索

访问顶点s

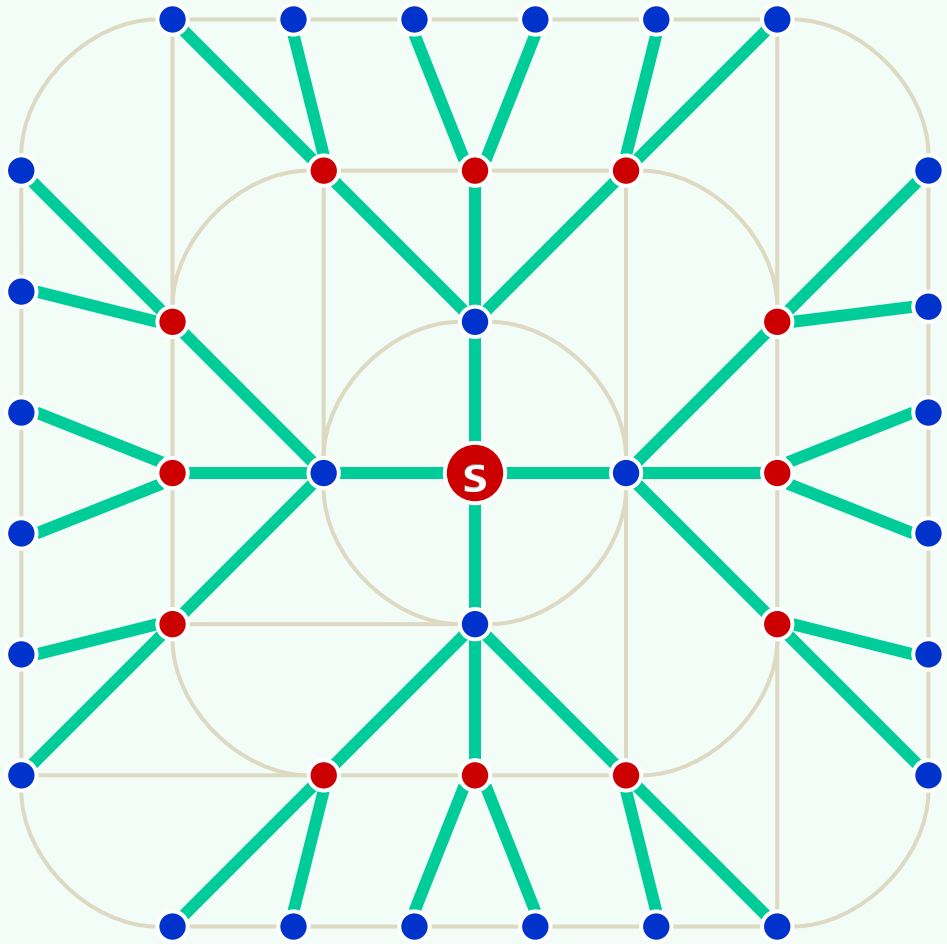
依次访问s所有尚未访问的邻接顶点

依次访问它们尚未访问的邻接顶点

...

如此反复

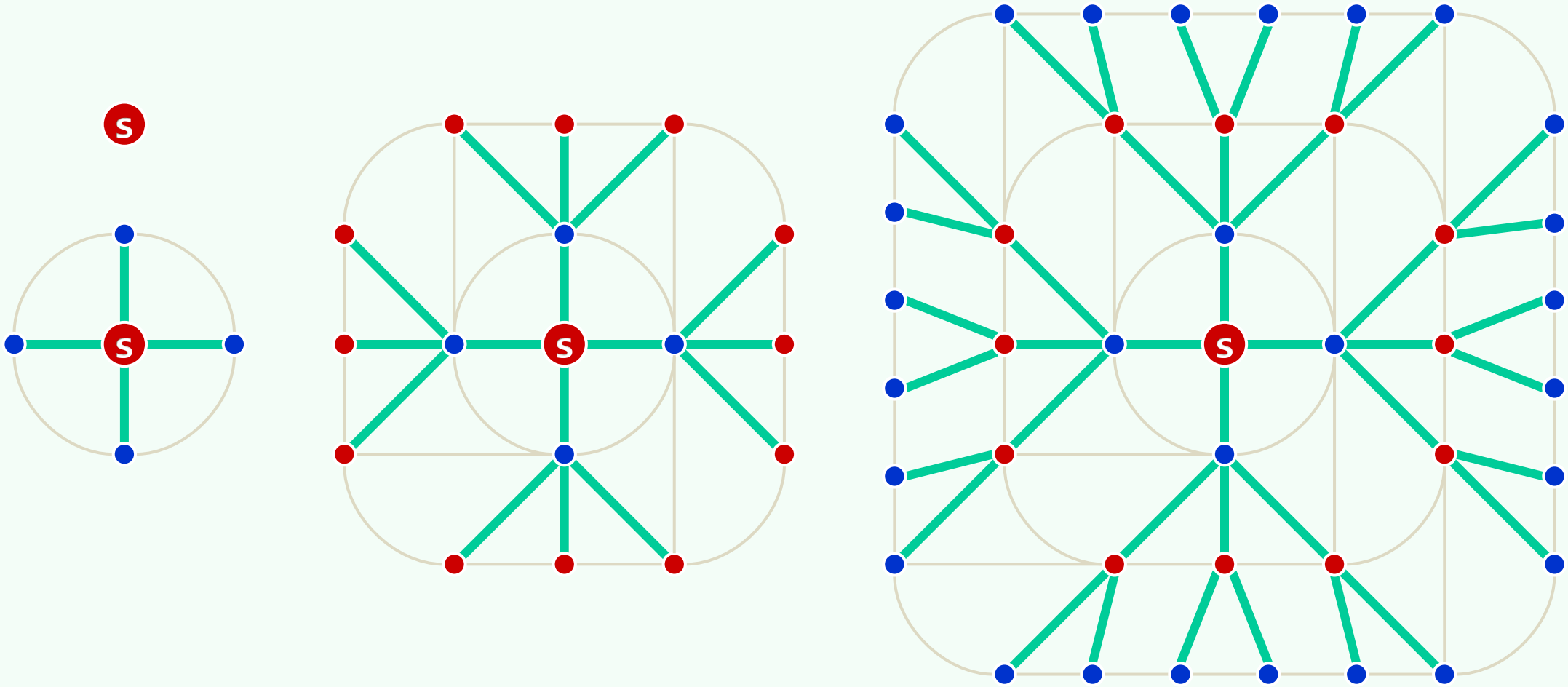
直至没有尚未访问的邻接顶点



❖ 以上策略及过程完全等同于树的层次遍历

❖ 事实上，BFS也的确会构造出原图的一棵支撑树（BFS tree）

譬喻：油气管线 + 绳网 + 涟漪



Graph::BFS() [1/2]

```
❖template <typename Tv, typename Te>
  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
  v     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 ... */
  v     status(v) = VISITED; //至此，当前顶点访问完毕
     }
  }
```

Graph::BFS() [2/2]

```
❖   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
    }
```