

图

广度优先搜索：算法

10-D1

邓俊辉

deng@tsinghua.edu.cn

我放下书，想，这么大一座园子，要在其中找到她的儿子，母亲走过了多少焦灼的路

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

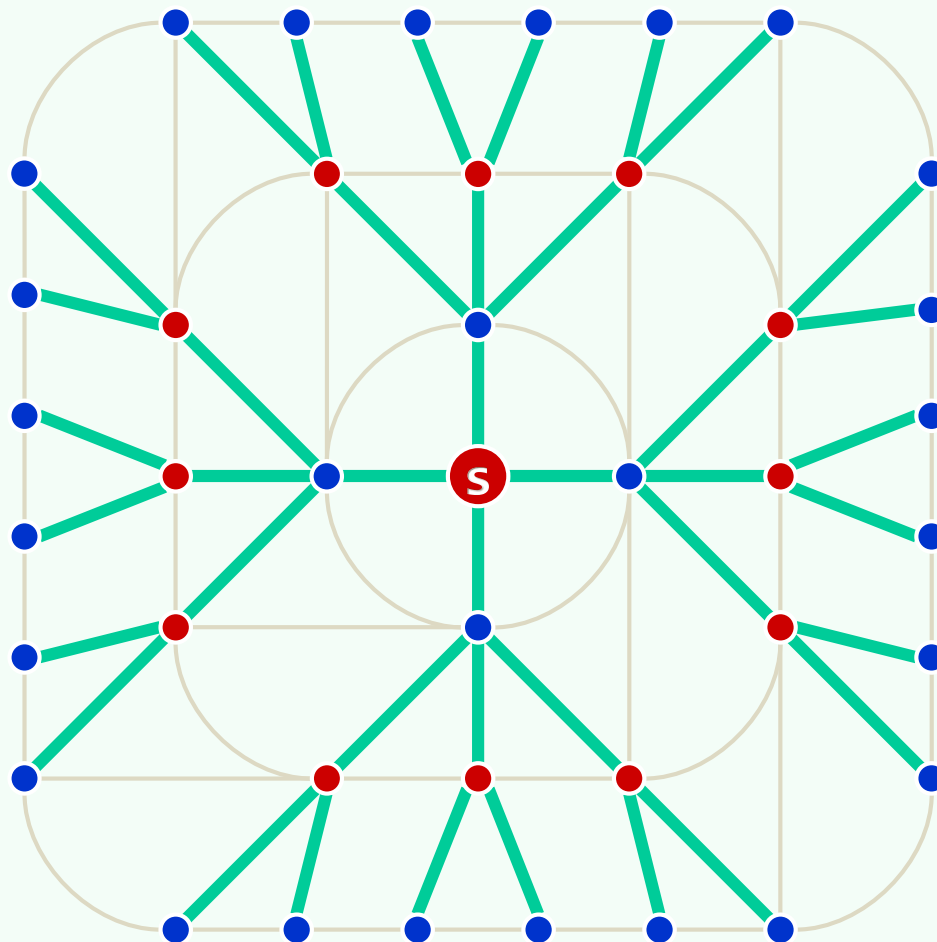
...

如此反复

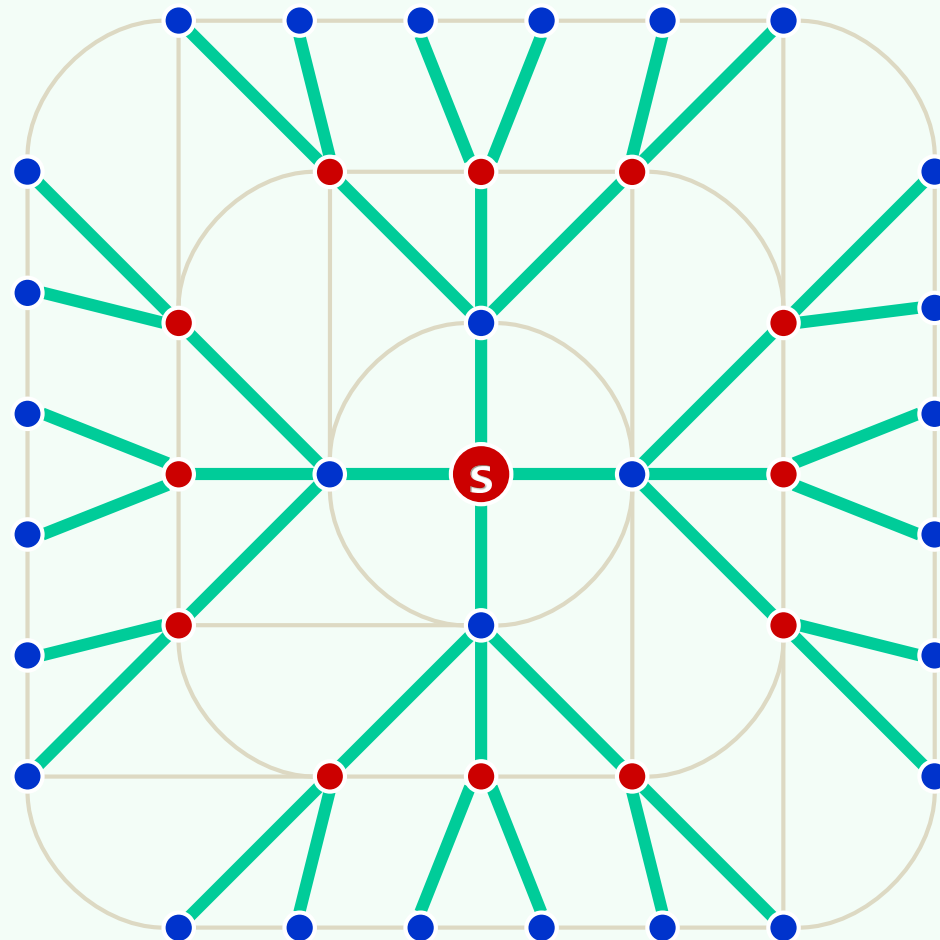
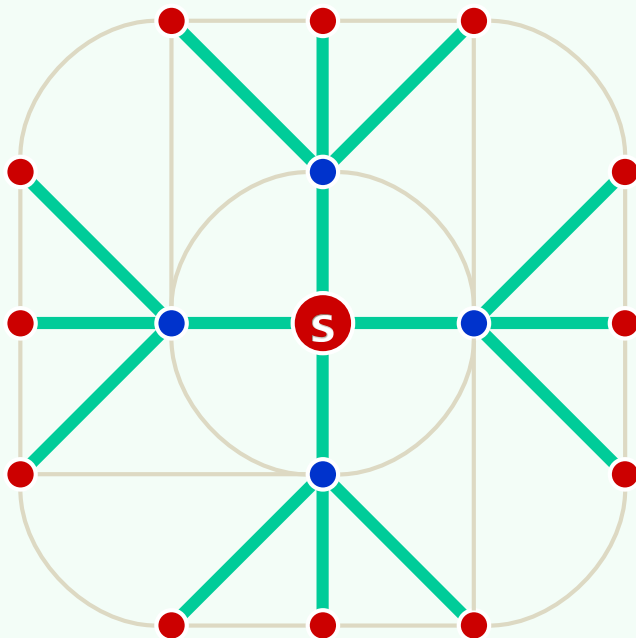
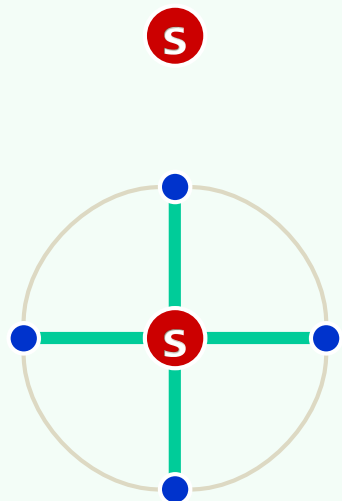
直至没有**尚未访问**的邻接顶点

❖ 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( Rank v, Rank& dClock ) {  
    Queue<Rank> Q; status(v) = DISCOVERED; Q.enqueue(v); dTime(v) = dClock++; //起点  
    for ( Rank fClock = 0; !Q.empty(); ) { //在Q变空之前, 反复地  
        if ( dTime(v) < dTime( Q.front() ) ) //dTime的增加, 意味着开启新一代, 因此  
            v dClock++, fClock = 0; //dTime递增, fTime复位  
        v = Q.dequeue(); //取出队首顶点v, 并  
        for ( Rank u = firstNbr(v); -1 != u; u = nextNbr(v, u) ) //考查v的每一个邻居u  
            v /* ... 视u的状态分别处理: 最终, 所有顶点按[dTime,fTime]字典序被遍历 ... */  
            status(v) = VISITED; fTime(v) = fClock++; //至此, v访问完毕  
    } //for  
} //BFS
```

Graph::BFS() [2/2]

```
/* ..... */
```

```
v v = Q.dequeue(); //取出队首顶点v, 并
```

```
for ( Rank u = firstNbr(v); -1 != u; u = nextNbr(v, u) ) //考查v的每一个邻居u
```

```
u if ( UNDISCOVERED == status(u) ) { //若u尚未被发现, 则发现之
```

```
u status(u) = DISCOVERED; Q.enqueue(u); dTime(u) = dClock; //发现该顶点
```

```
type(v, u) = TREE; parent(u) = v; //引入树边, 拓展BFS树
```

```
u } 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
```

u

```
type(v, u) = CROSS; //将(v, u)归类于跨边
```

```
v status(v) = VISITED; fTime(v) = fClock++; //至此, v访问完毕
```

```
/* ..... */
```