

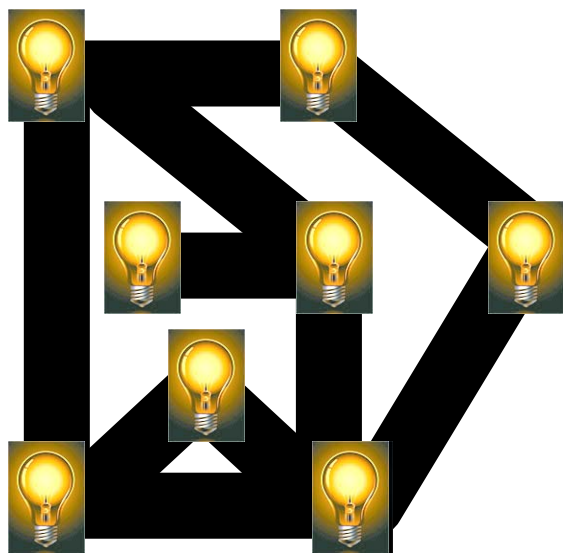
# 第六讲 图（上）

浙江大学 陈 越

## 6.2 图的遍历

# 深度优先搜索 (Depth First Search, DFS)

类似于树的先序遍历

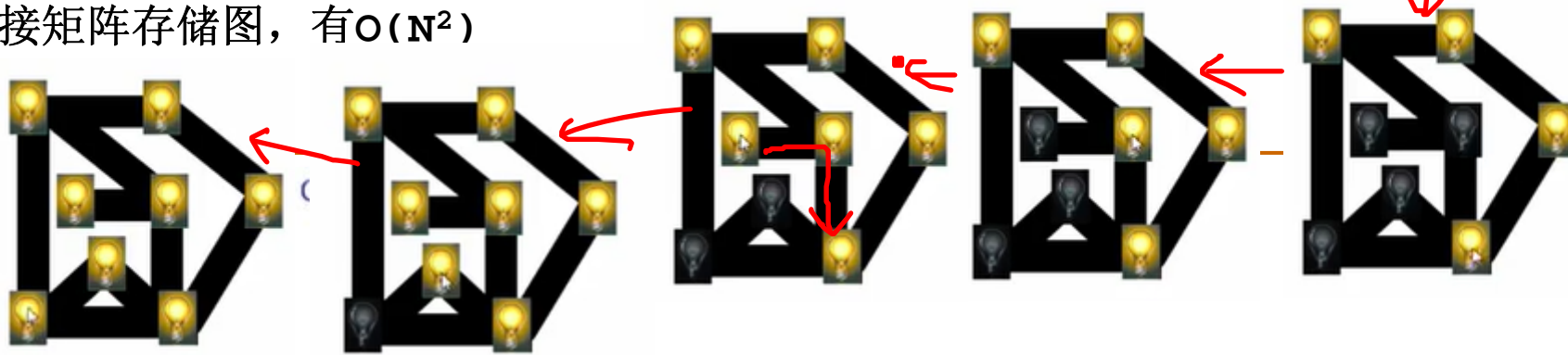
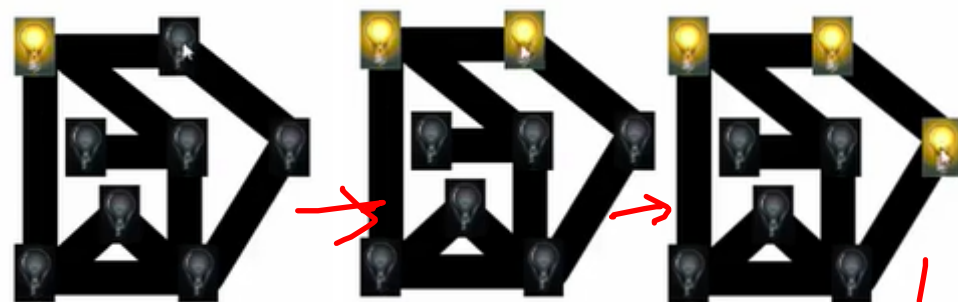


```
void DFS ( Vertex V )
{
    visited[ V ] = true;
    for ( v 的每个邻接点 W )
        if ( !visited[ W ] )
            DFS( W );
}
```

若有 $N$ 个顶点、 $E$ 条边，时间复杂度是

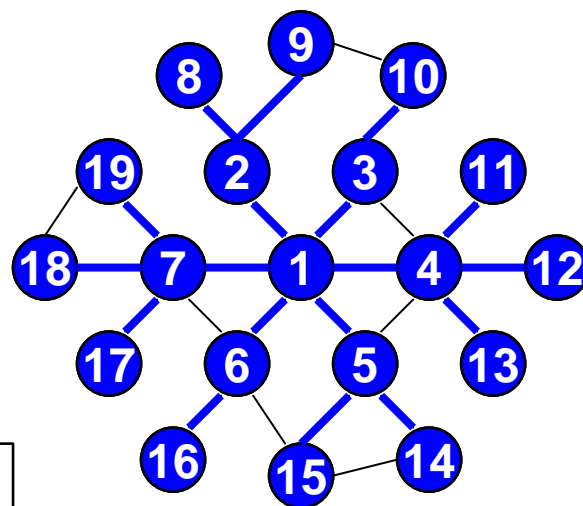
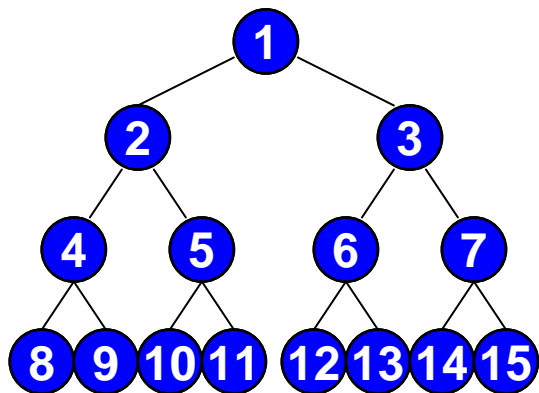
- 用邻接表存储图，有 $O(N+E)$
- 用邻接矩阵存储图，有 $O(N^2)$

因为需要访问一行里的所有值，才能知道哪个是他的邻接点，所以需要 $N*N$



# 广度优先搜索 (Breadth First Search, BFS)

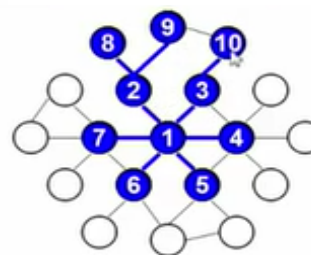
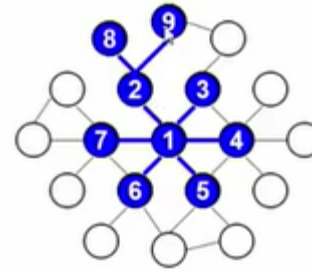
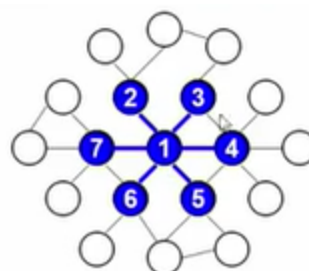
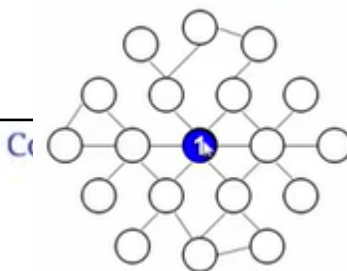
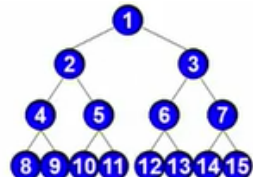
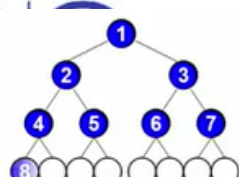
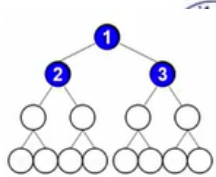
层序遍历



```
void BFS ( Vertex V )  
{ visited[V] = true;  
  Enqueue(V, Q);  
  while(!IsEmpty(Q)){  
    V = Dequeue(Q);  
    for ( V 的每个邻接点 W )  
      if ( !visited[W] ) {  
        visited[W] = true;  
        Enqueue(W, Q);  
      }  
  }  
}
```

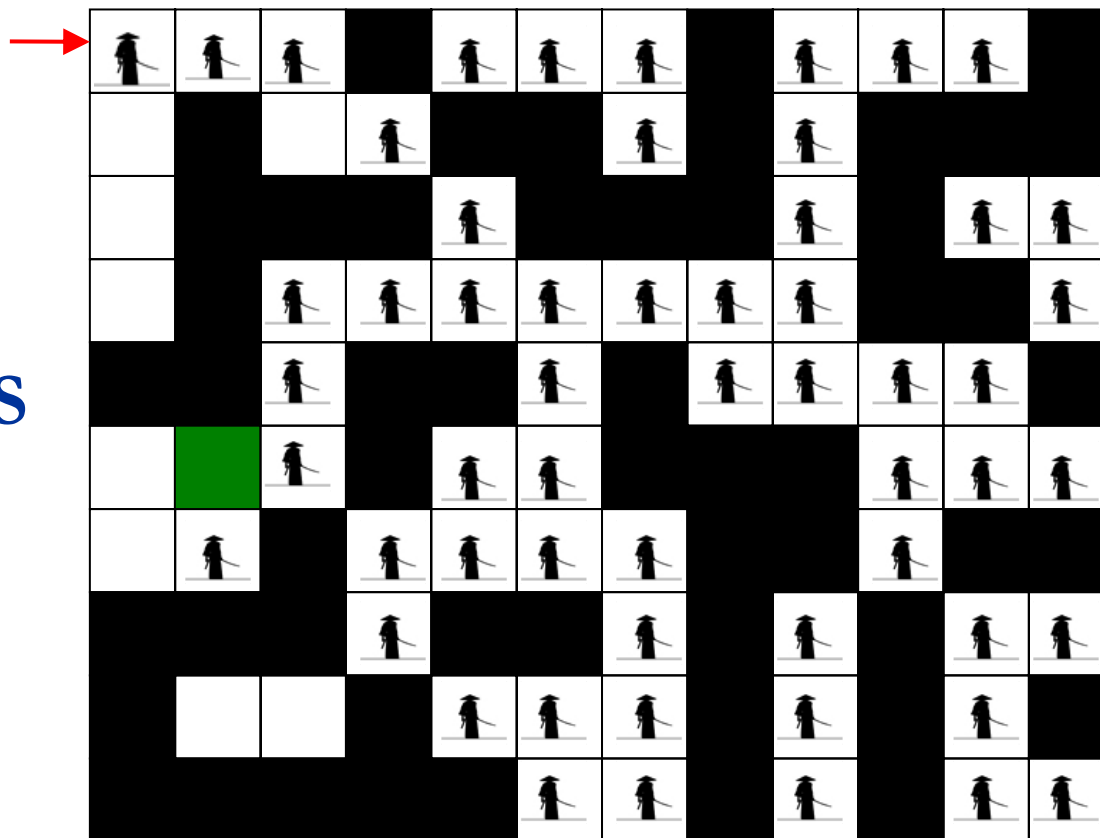
若有 $N$ 个顶点、 $E$ 条边，时间复杂度是

- 用邻接表存储图，有 $O(N+E)$
- 用邻接矩阵存储图，有 $O(N^2)$



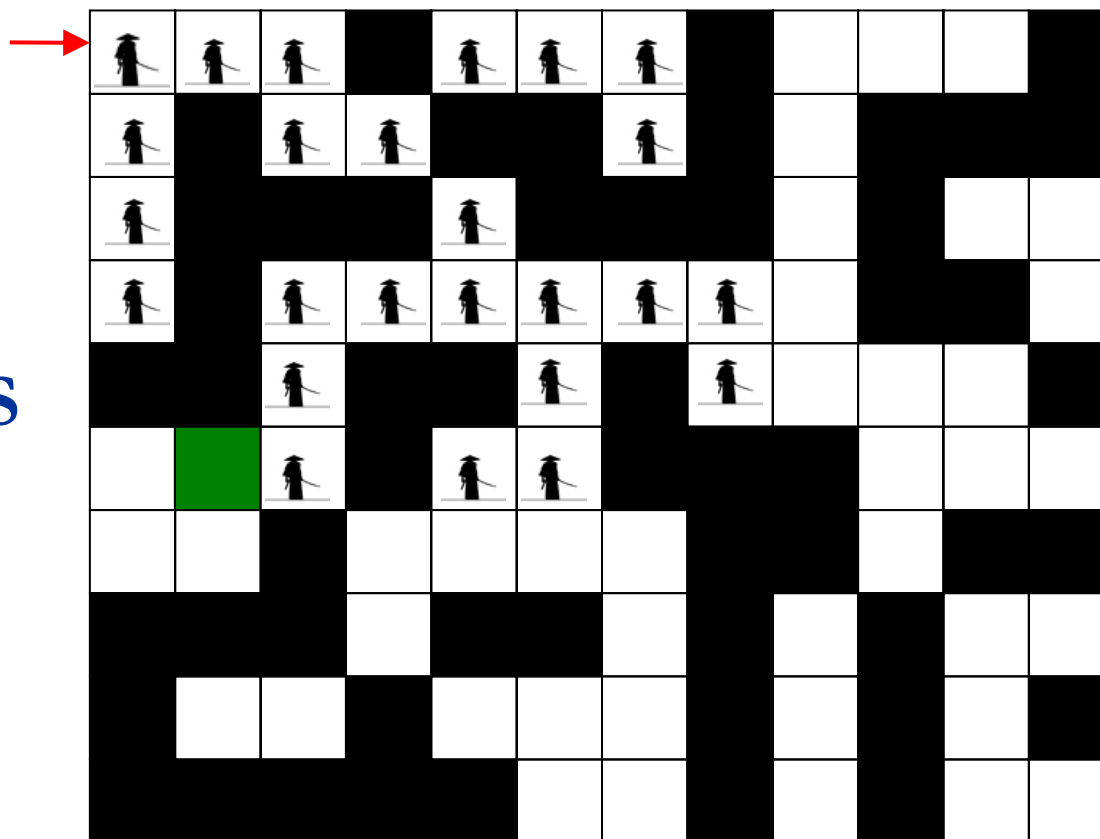
# 为什么需要两种遍历？

DFS

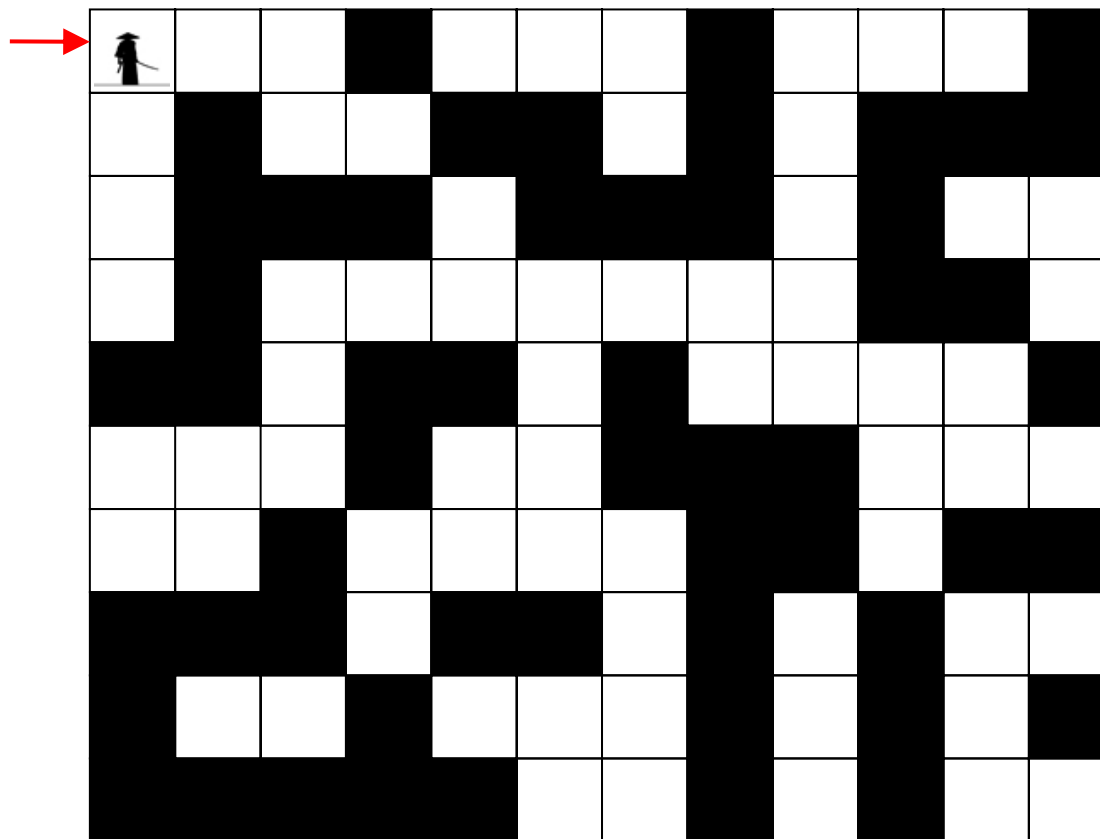


# 为什么需要两种遍历？

BFS



# 为什么需要两种遍历？



把出口换到哪里就该BFS不爽了？

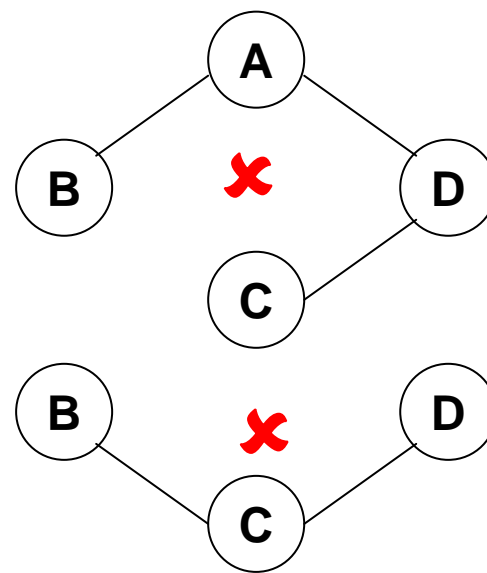
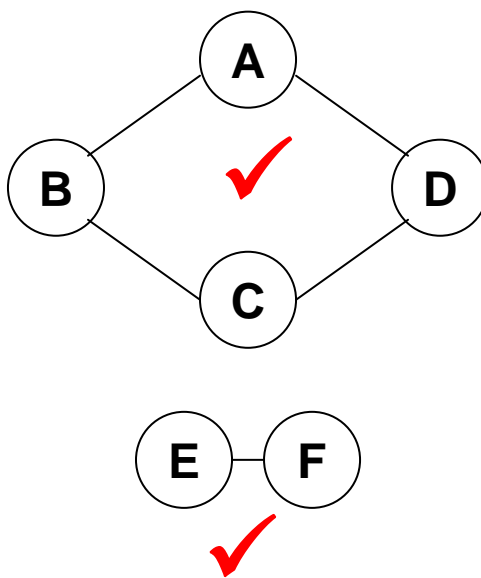
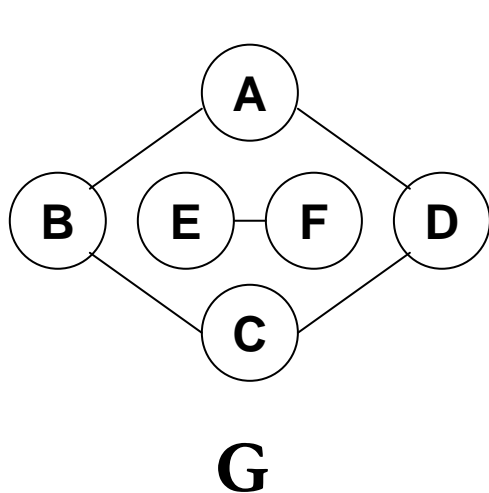
# 图不连通怎么办？

- **连通**：如果从 $v$ 到 $w$ 存在一条（无向）**路径**，则称 $v$ 和 $w$ 是连通的
- **路径**： $v$ 到 $w$ 的路径是一系列顶点 $\{v, v_1, v_2, \dots, v_n, w\}$ 的集合，其中任一对相邻的顶点间都有图中的边。**路径的长度**是路径中的边数（如果带权，则是所有边的权重和）。如果 $v$ 到 $w$ 之间的所有顶点都不同，则称**简单路径**
- **回路**：起点等于终点的路径
- **连通图**：图中任意两顶点均连通



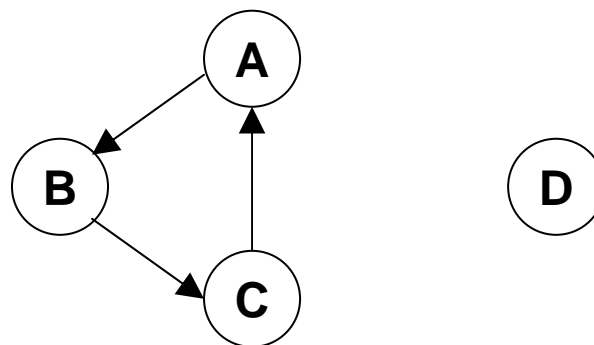
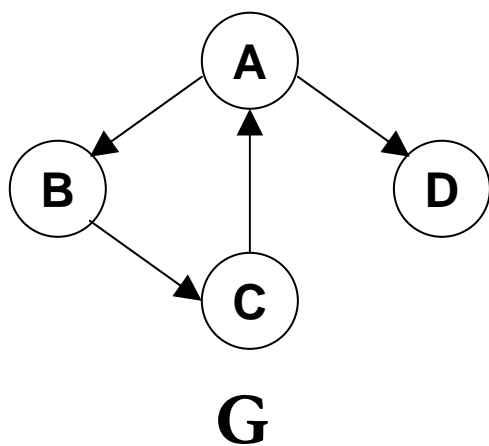
# 图不连通怎么办？

- **连通分量**：无向图的**极大**连通子图
  - 极大顶点数：再加1个顶点就不连通了
  - 极大边数：包含子图中所有顶点相连的所有边



# 图不连通怎么办？

- **强连通**：有向图中顶点 $v$ 和 $w$ 之间存在双向路径，则称 $v$ 和 $w$ 是强连通的
- **强连通图**：有向图中任意两顶点均强连通
- **强连通分量**：有向图的极大强连通子图



# 图不连通怎么办？

```
void DFS ( Vertex V )
{ visited[ V ] = true;
  for ( V 的每个邻接点 W )
    if ( !visited[ W ] )
      DFS( W );
}
```

每调用一次DFS(V)，就把V所在的连通分量遍历了一遍。BFS也是一样。

```
void ListComponents ( Graph G )
{ for ( each V in G )
  if ( !visited[V] ) {
    DFS( V ); /*or BFS( V )*/
  }
}
```