03.2 - Depth-First Search

DFS

Implementing DFS

实现深度优先搜索（DFS）时，我们需要满足以下性质：

- 我们可以在 $O(\deg(v))$ 时间内找到所有与顶点 v 相连的边
- 给定边 e 的一个端点，我们可以在 $O(1)$ 时间内找到该边的另一个端点

上面的两条性质正好可以由邻接表来表示，并且

- 我们需要一种方法来标记节点或边为“已探索”，并且能够在 $O(1)$ 时间内测试某个节点或边是否已经被探索。换句话说，我们在遍历过程中不会重复检查同一条边

由上所述，我们需要以下的数据结构：

- 邻接表：用于存储图的结构
  - 每个顶点都有一个 `.next` 指针，用于按照出现顺序遍历该顶点的邻居节点。例如 `u.next` 可以获取顶点 u 的下一个邻居节点
- 栈：用于存储在 DFS 过程中需要访问的节点
  - 栈是一种后进先出的数据结构，元素按照顺序被压入栈顶，并且从栈顶弹出
- 数组：一个布尔数组 `explored[1, ..., n]`，同于存储每个节点是否已被探索

DFS 算法与伪代码

1. 随机从一个节点 s 开始，并将 s 标记为“已访问”
2. 用 u 表示当前节点，初始化 $u = s$
3. 沿任意边 (u, v) 旅行
   1. 如果 (u, v) 通向一个已访问过的顶点，返回到 u
   2. 否则，将 v 标记为已访问，并设置 $u = v$
   3. 返回步骤 3
4. 当我们到达一个死胡同（所有邻居都已访问），回溯到前一个访问的顶点 p，并设置 $u = p$，返回步骤 3
5. 终止：回溯到 s 时，终止

```
// 递归版本
Algorithm DFS(G, v)
    for all edges e incident to v:  // 所有与顶点 v 相连的边
        if edge e is unexplored
            Let u be the other endpoint of e  // 获取 e 的另一个端点 u
            If vertex u is unexplored
                Label e as a discovery edge
                DFS(G, u)
            Else
                Label e as a back edge
```

```
// 迭代版本（栈）
DFS_Iterative(start_node):
    1. 创建一个空栈 stack
    2. 将 start_node 压入 stack
    3. 标记 start_node 为已访问

    4. 当 stack 不为空:
        5. 弹出 stack 顶部的节点 node
        6. 对于每一个 node 的邻居 neighbor:
            7. 如果 neighbor 未被访问:
                8. 将 neighbor 压入 stack
                9. 标记 neighbor 为已访问
// stack为空时表示整张图都被检查过了
```

Properties of DFS

- 为了简化讨论，假设图是连通的。这意味着图中任意两个顶点之间都有路径相连
- DFS会访问图中的所有节点
  - 假设存在某个节点 v 未被访问，并且设 w 是从起始节点 s 到节点 v 的某条路径上的第一个未访问的节点
  - 由于 w 是第一个未访问的节点，则 w 的某个邻居 u 必定已经被访问
  - 这样，边 (u, w) 应该已经被探索，而 w 应该已经被访问。因此，假设不成立

- DFS过程中标记的发现边（discovery edges）会形成一棵生成树
  - 我们只在访问未访问的节点时将边标记为发现边，因此发现边不会形成环

DFS 的时间复杂度

- 每个节点在DFS过程中只被访问一次
- 每条边恰好被检查两次
  - 每条边从其每个端点节点都检查一次。例如，对于边 (u, v)，它会在访问 u 时检查一次，在访问 v 时再次检查
- 因此，DFS 的时间复杂度是 $O(n + m)$

Spanning Tree & Connected Component

生成树（Spanning Tree）

一个图的生成树是该图的一个子图，包含图中所有的顶点，并且是一个树（即无环连通图）

- 子图：生成树包含原图中的所有顶点，但只包含部分边
- 连通性：生成树必须连通，也就是说，从任意一个顶点到其他任意一个顶点都有路径
- 无环性：生成树不能有环

而于一个有 n 个顶点的连通无向图，其生成树有以下性质：

- 包含 $n - 1$ 条边
- 如果生成树中有 n 条边，它必定有环
- 如果生成树的边小于 $n - 1$ 条，它必定不联通

连通分量（Connected Component）

连通图：如果图中任意两点都是连通的，那么图被称作连通图

连通子图是图中一个子集，它满足

- 连通子图中的任意两个顶点之间都存在路径
- 连通子图包含原图中的一部分顶点和边

极大联通子图（Maximal Connected Subgraph）：是指在无向图中，不能通过添加任何更多顶点或边而仍然保持连通的连通子图。一个图中可以有多个极大连通子图（这多个子图互不相连）

```
A - B       D - E
     \      |
      C     F

G - H - I
```

连通分量（Connected Component）是指无向图中一个极大连通子图，每一个顶点和每一条边都属于唯一的一个连通分量，连通图只有一个连通分量，即其自身；非连通的无向图有多个连通分量