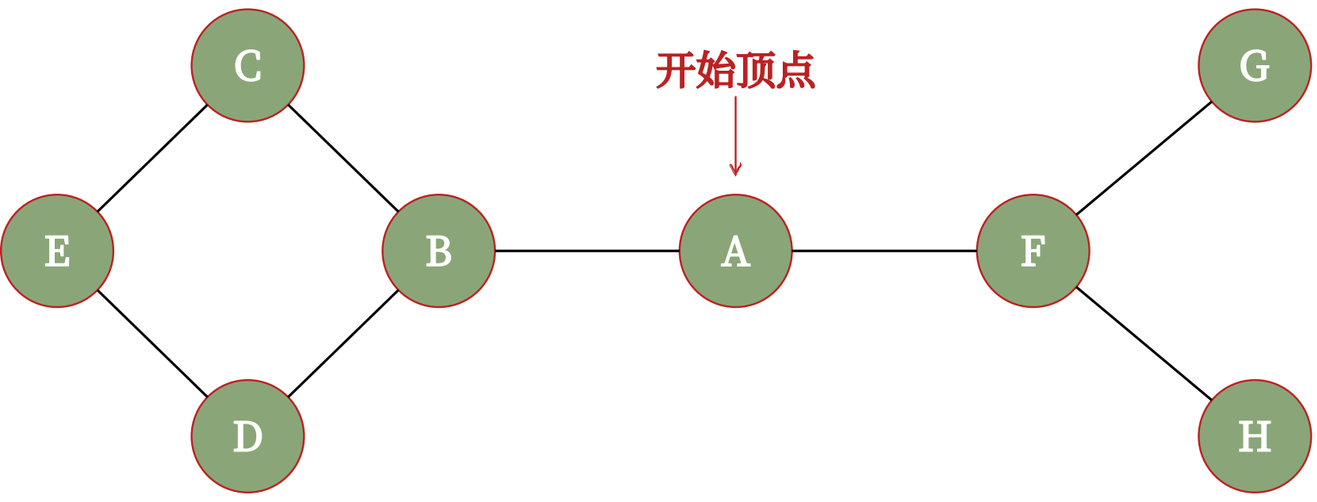

深度优先遍历非递归算法

Input: 图 $G = (V, E)$，起始顶点 s

Output: 深度优先遍历的访问序列

```
创建一个空的集合 visited 用于存储已访问的顶点;
创建一个空的栈 stack 并将起始顶点 s 入栈;
while stack 非空 do
    弹出栈顶元素 v;
    if v 不在 visited 中 then
        将 v 标记为已访问，即加入到 visited;
        输出顶点 v;
        for 邻接顶点 w of v （从后向前遍历）do
            if w 不在 visited 中 then
                将 w 压入栈 stack;
            end
        end
    end
end
```

阅读材料: Tarjan, R., 1971. Depth-first search and linear graph algorithms

初始状态:
visited = []
stack = ['A']

第一次 while 循环
从栈中弹出 'A'，由于'A'不在 visited 中，加入到 visited， visited= ['A']
第一次 for 循环：
检查'A'的邻接顶点'F'，由于'F'不在visited中，将其压入栈中，stack = ['F']。
第二次 for 循环：
检查'A'的邻接顶点'B'，由于'B'不在visited中，将其压入栈中，stack = ['F','B']。
visited= ['A']
stack = ['F','B']。

第二次 while 循环
从栈中弹出 'B'，由于'B'不在 visited 中，加入到 visited， visited= ['A', 'B']
第一次 for 循环：
检查'B'的邻接顶点'D'，由于'D'不在visited中，将其压入栈中，stack = ['F', 'D']。
第二次 for 循环：
检查'B'的邻接顶点'C'，由于'C'不在visited中，将其压入栈中，stack = ['F','D', 'C']。
第三次 for 循环：
检查'B'的邻接顶点'A'，由于'A'在visited中，停止。
visited= ['A', 'B']
stack = ['F','D', 'C']。

第三次 while 循环
从栈中弹出 'C'，由于'C'不在 visited 中，加入到 visited， visited= ['A', 'B', 'C']
第一次 for 循环：
检查'C'的邻接顶点'E'，由于'E'不在visited中，将其压入栈中，stack = ['F', 'D', 'E']。
第二次 for 循环：
检查'C'的邻接顶点'B'，由于'B'在visited中，停止。
visited= ['A', 'B', 'C']
stack = ['F', 'D', 'E']

第四次 while 循环
从栈中弹出 'E'，由于'E'不在 visited 中，加入到 visited， visited= ['A', 'B', 'C', 'E']
第一次 for 循环：
检查'E'的邻接顶点'D'，由于'D'不在visited中，将其压入栈中，stack = ['F', 'D', 'D']。
第二次 for 循环：
检查'E'的邻接顶点'C'，由于'C'在visited中，停止。
visited= ['A', 'B', 'C', 'E']
stack = ['F', 'D', 'D']

第五次 while 循环
从栈中弹出 'D'，由于'D'不在 visited 中，加入到 visited， visited= ['A', 'B', 'C', 'E', 'D']
第一次 for 循环：
检查'D'的邻接顶点'E'，由于'E'在visited中，停止。
第二次 for 循环：
检查'D'的邻接顶点'B'，由于'B'在visited中，停止。
visited= ['A', 'B', 'C', 'E', 'D']
stack = ['F', 'D']

第六次 while 循环
从栈中弹出 'D'，由于'D'在 visited 中，停止
visited= ['A', 'B', 'C', 'E', 'D']
stack = ['F']

第七次 while 循环
从栈中弹出 'F'，由于'F'不在 visited 中，加入到 visited， visited= ['A', 'B', 'C', 'E', 'D', 'F']
第一次 for 循环：
检查'F'的邻接顶点'H'，由于'H'不在visited中，将其压入栈中，stack = ['H']。
第二次 for 循环：
检查'F'的邻接顶点'G'，由于'G'不在visited中，将其压入栈中，stack = ['H', 'G']。
visited= ['A', 'B', 'C', 'E', 'D', 'F']
stack = ['H', 'G']。

第八次 while 循环
从栈中弹出 'G'，由于'G'不在 visited 中，加入到 visited， visited= ['A', 'B', 'C', 'E', 'D', 'F', 'G']
第一次 for 循环：
检查'G'的邻接顶点'F'，由于'F'在visited中，停止。
visited= ['A', 'B', 'C', 'E', 'D', 'F', 'G']
stack = ['H']

第九次 while 循环
从栈中弹出 'H'，由于'H'不在 visited 中，加入到 visited， visited= ['A', 'B', 'C', 'E', 'D', 'F', 'G', 'H']
第一次 for 循环：
检查'H'的邻接顶点'F'，由于'F'在visited中，停止。
visited= ['A', 'B', 'C', 'E', 'D', 'F', 'G', 'H']
stack = []

由于栈已空，停止循环，深度优先遍历的序列为：['A', 'B', 'C', 'E', 'D', 'F', 'G', 'H']

Copyright ©ZHANG Luping