Design and Analysis of Algorithms

Part IV: Graph Algorithms

Lecture 24: Cycle Detection

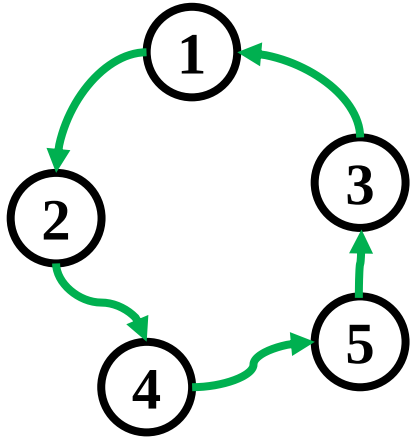
童咏昕

北京航空航天大学
计算机学院

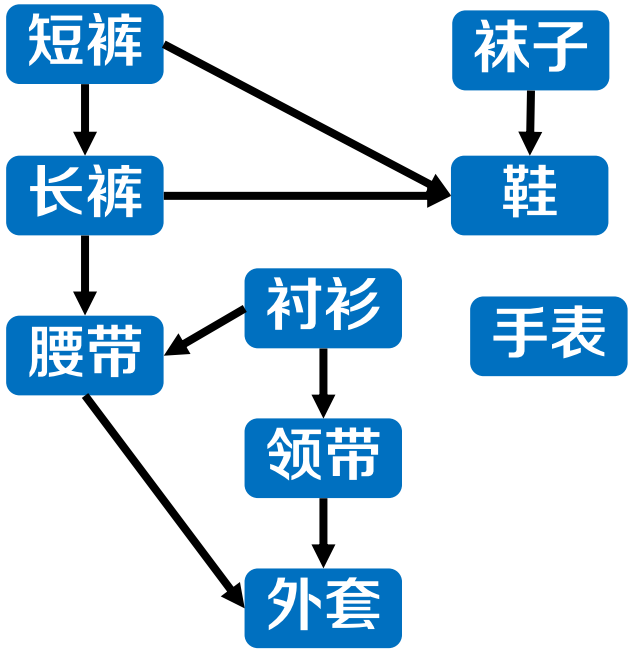
图算法篇概述

- 在算法课程第四部分“图算法”主题中，我们将主要聚焦于如下经典问题：
  - Basic Concepts in Graph Algorithms（图算法的基本概念）
  - Breadth-First Search (BFS，广度优先搜索）
  - Depth-First Search (DFS，深度优先搜索）
  - Cycle Detection（环路检测）
  - Topological Sort（拓扑排序）
  - Strongly Connected Components（强连通分量）
  - Minimum Spanning Trees (最小生成树)
  - Single Source Shortest Path (单源最短路径)
  - All-Pairs Shortest Paths (所有点对最短路径)
  - Bipartite Graph Matching（二分图匹配）
  - Maximum/Network Flows (最大流/网络流)

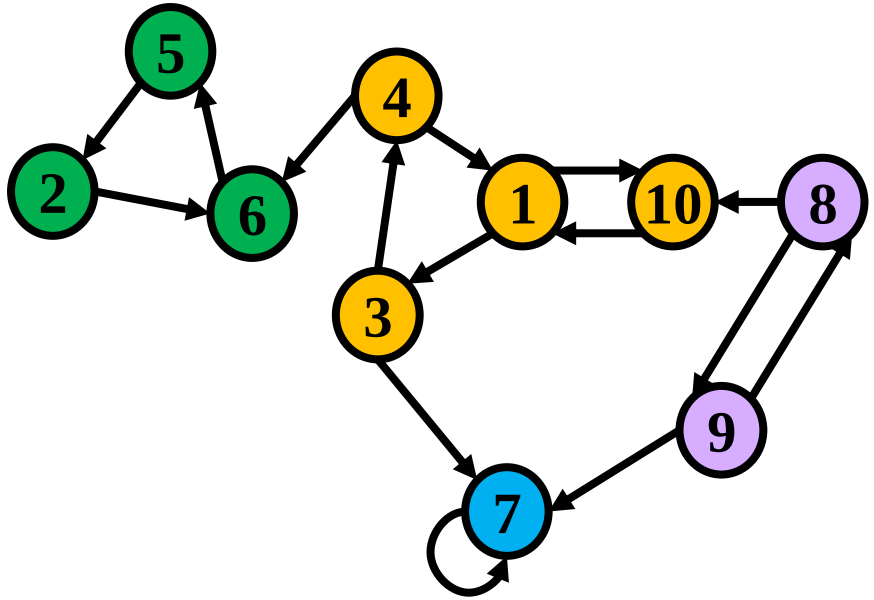
深度优先搜索应用



环路的存在性判断



拓扑排序



强连通分量

问题定义

有向图中环路的存在性判断

输入

- 有向图$G =< V, E >$，V是顶点集合，E是边的集合

输出

- 图G是否存在环

问题定义

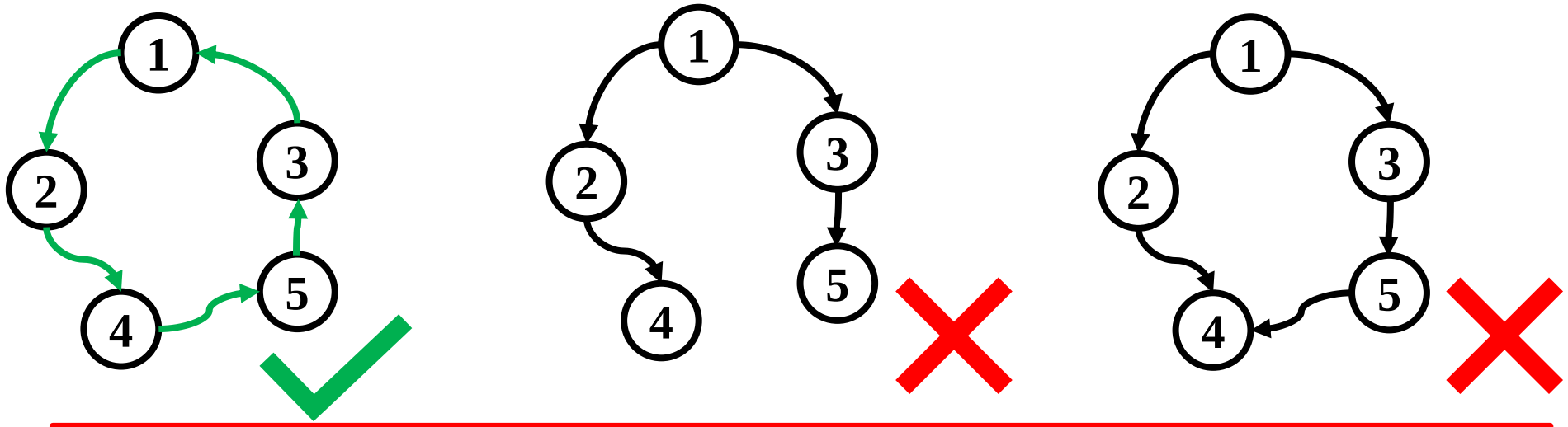
有向图中环路的存在性判断

输入

- 有向图$G=<V,E>$，V是顶点集合，E是边的集合

输出

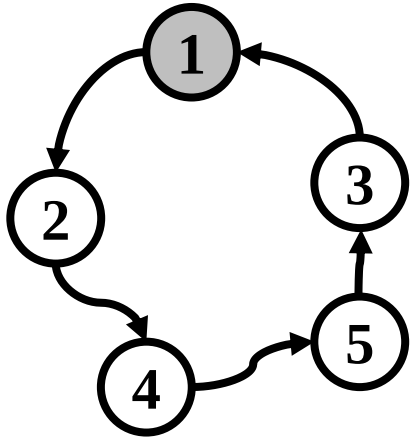
- 图G是否存在环



问题：深度优先搜索边的性质能否帮助解决问题?

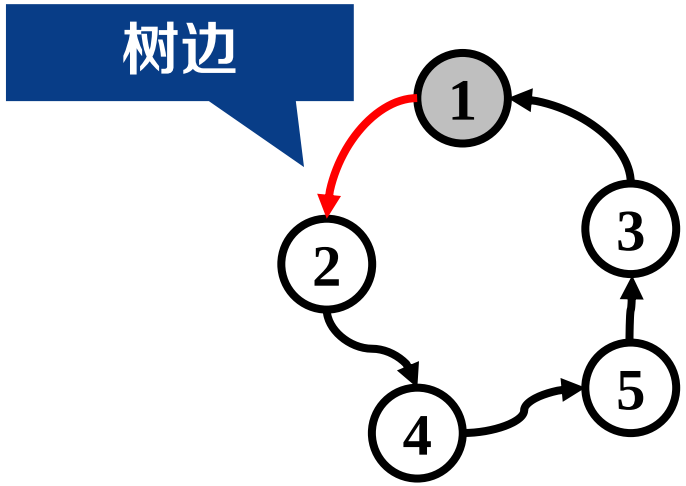
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



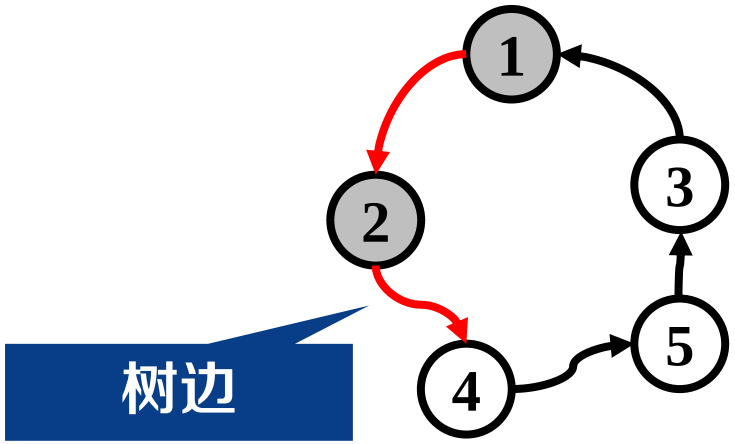
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



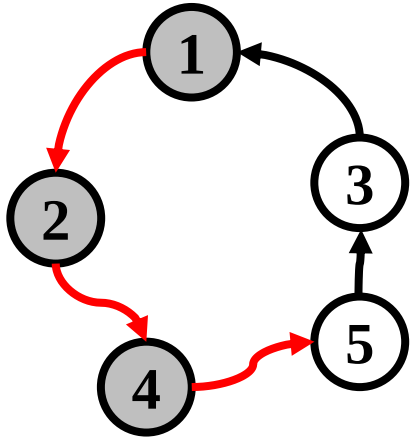
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



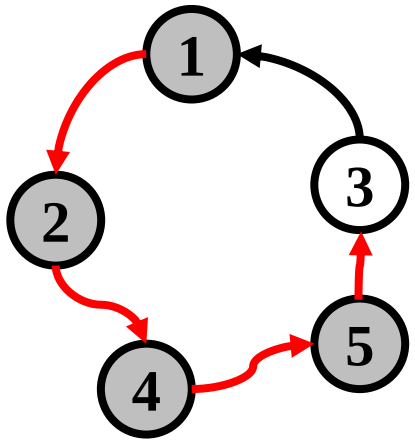
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



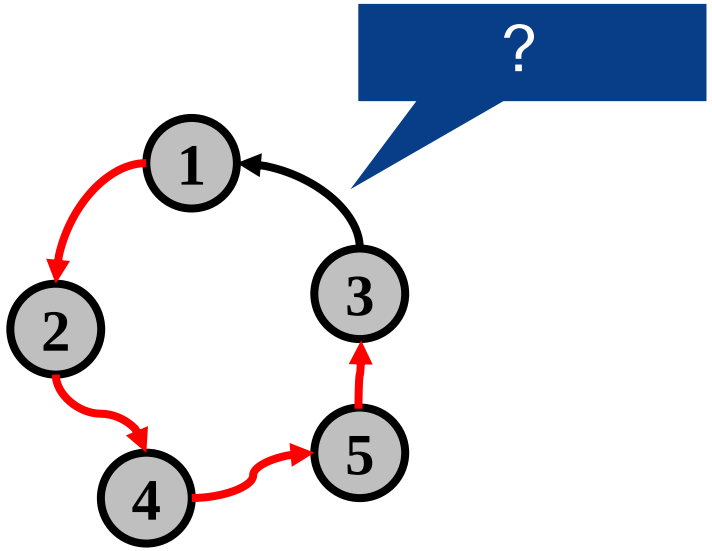
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



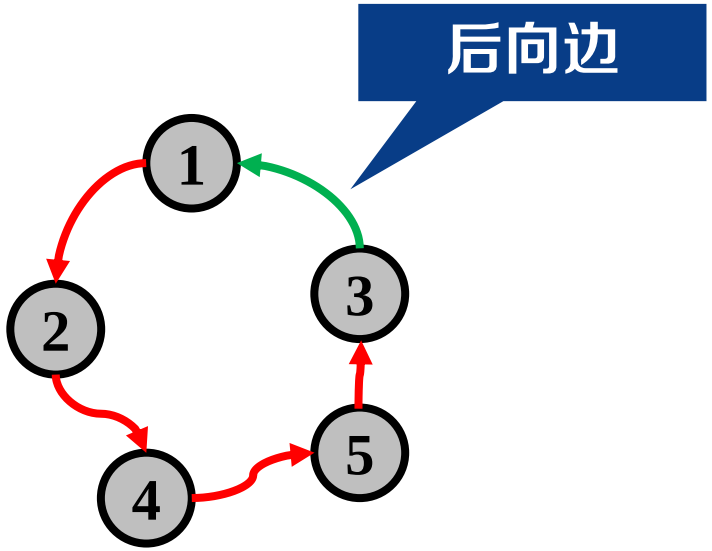
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



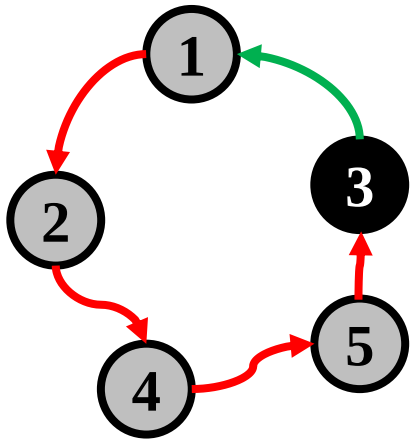
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



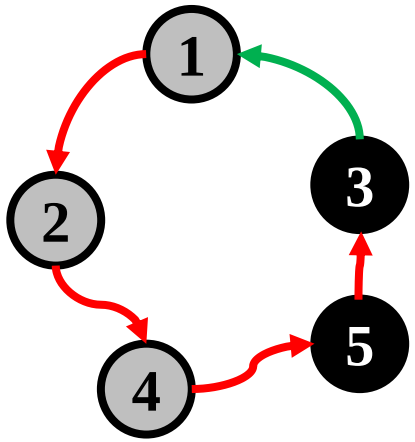
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



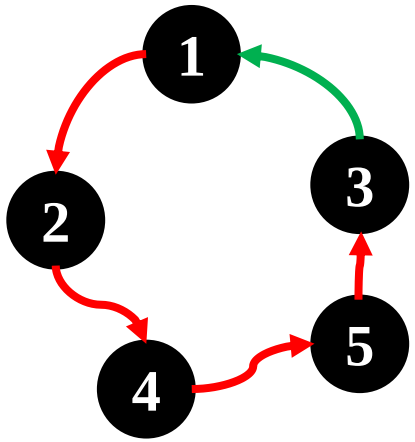
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



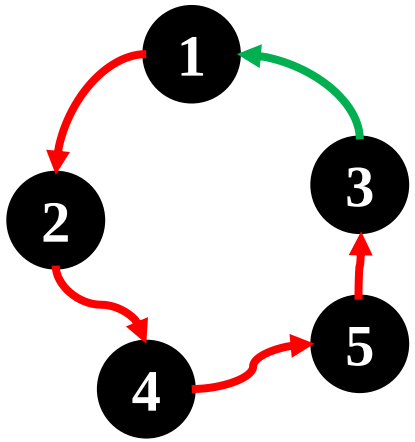
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边



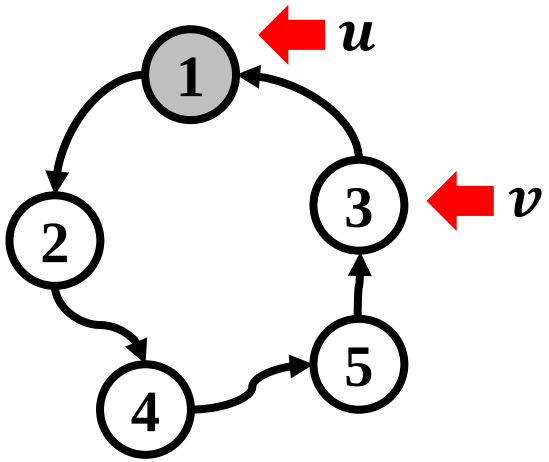
问题观察

- 观察环路上深度优先搜索时边的种类
  - 树边：在深度优先树中的边
  - 前向边：不在深度优先树中，从祖先指向后代的边
  - 后向边：从后代指向祖先的边
  - 横向边：顶点不具有祖先后代关系的边
- 猜想：有向图存在环路⇔搜索时出现后向边



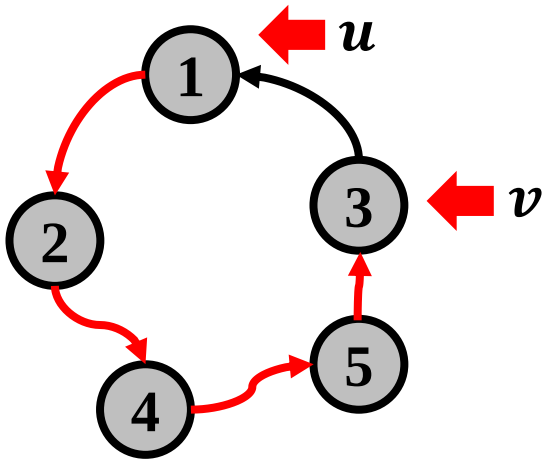
猜想证明

- 猜想：有向图存在环路⇔搜索时出现后向边
- 证明：有向图存在环路⇒搜索时出现后向边
  - 不妨设环路上被搜索的第一个点为u，v是在环路上指向u的点



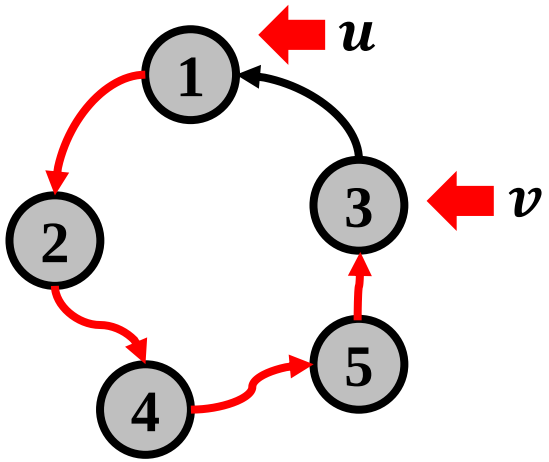
猜想证明

- 猜想：有向图存在环路⇔搜索时出现后向边
- 证明：有向图存在环路⇒搜索时出现后向边
  - 不妨设环路上被搜索的第一个点为u，v是在环路上指向u的点
  - u可达v，深度优先搜索可以搜索到v



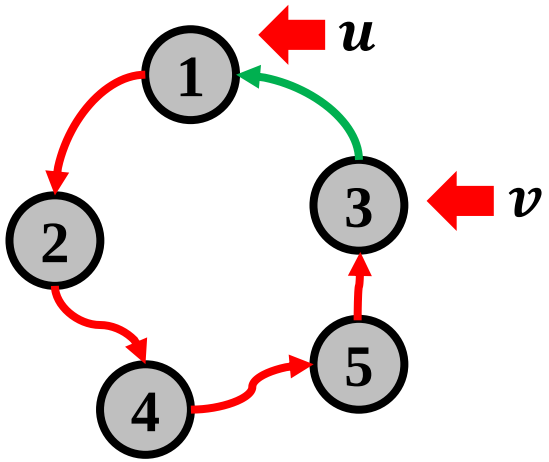
猜想证明

- 猜想：有向图存在环路⇔搜索时出现后向边
- 证明：有向图存在环路⇒搜索时出现后向边
  - 不妨设环路上被搜索的第一个点为u，v是在环路上指向u的点
  - u可达v，深度优先搜索可以搜索到v
  - 搜索v时，由于v指向u，必能再次发现顶点u



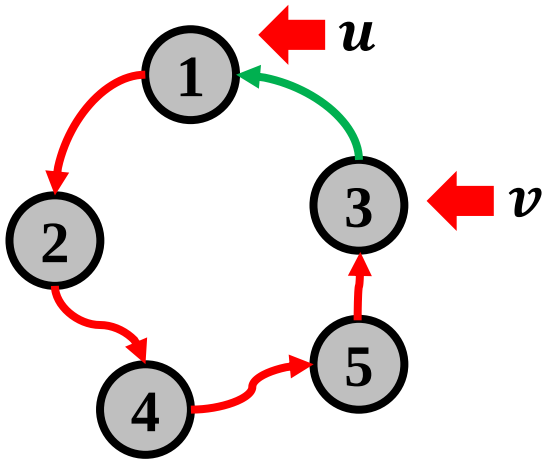
猜想证明

- 猜想：有向图存在环路⇔搜索时出现后向边
- 证明：有向图存在环路⇒搜索时出现后向边
  - 不妨设环路上被搜索的第一个点为u，v是在环路上指向u的点
  - u可达v，深度优先搜索可以搜索到v
  - 搜索v时，由于v指向u，必能再次发现顶点u
  - 从后代搜索祖先，出现后向边



猜想证明

- 猜想：有向图存在环路⇔搜索时出现后向边
- 证明：有向图存在环路⇐搜索时出现后向边
  - 深度优先树中祖先可达后代
  - 后向边从后代指向祖先
  - 后代和祖先之间存在环路



伪代码

- **DFS-Judge-Cycle(G)**

```
输入: 图G
输出: 是否存在环路
新建数组 color[1..V], pred[1..V]
//初始化
for v ∈ V do
    pred[v] ← NULL
    color[v] ← WHITE
end
for v ∈ V do
    if color[v] = WHITE then
        if DFS-Visit-Judge-Cycle(G, v) = TRUE then
            return TRUE
        end
    end
end
return FALSE
```

发现环则返回True

伪代码

- **DFS-Visit-Judge-Cycle(G, v)**

输入: 图G, 顶点v
输出: 顶点v是否在某环路中
$color[v] \leftarrow GRAY$
for $w \in G.Adj[v]$ **do**
  if $color[w] = GRAY$ **then**
    return $TRUE$
  end
  if $color[w] = WHITE$ **then**
    $pred[w] \leftarrow v$
    if $DFS\text{-}Visit\text{-}Judge\text{-}Cycle(G, w)$=$TRUE$ **then**
      return $TRUE$
    end
  end
end
$color[v] \leftarrow BLACK$
return $FALSE$

搜索到灰色点
（发现后向边）

伪代码

- **DFS-Visit-Judge-Cycle(G, v)**

```
输入: 图G, 顶点v
输出: 顶点v是否在某环路中
color[v] ← GRAY
for w ∈ G.Adj[v] do
    if color[w] = GRAY then
        return TRUE
    end
    if color[w] = WHITE then
        pred[w] ← v
        if DFS-Visit-Judge-Cycle(G, w)=TRUE then
            return TRUE
        end
    end
end
color[v] ← BLACK
return FALSE
```

递归发现环

复杂度分析

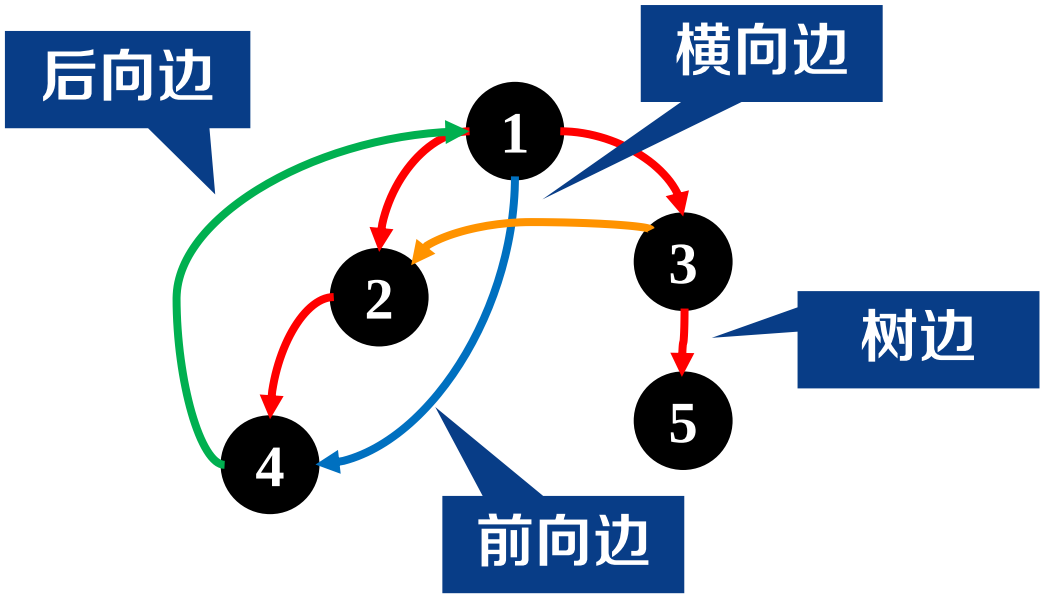
- **DFS-Visit-Judge-Cycle(G, v)**

```
输入: 图G, 顶点v
输出: 顶点v是否在某环路中
color[v] ← GRAY
for w ∈ G.Adj[v] do
    if color[w] = GRAY then
        return TRUE
    end
    if color[w] = WHITE then
        pred[w] ← v
        if DFS-Visit-Judge-Cycle(G, w)=TRUE then
            return TRUE
        end
    end
end
color[v] ← BLACK
return FALSE
```

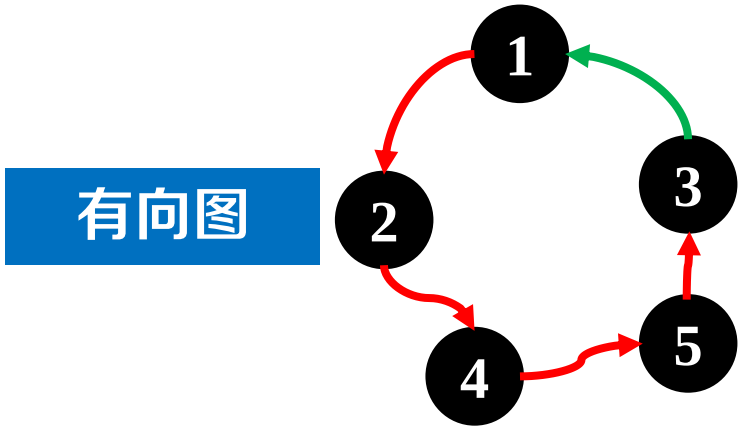
时间复杂度：$O(|V| + |E|)$

小结

- 有向图深度优先树中边的分类



- 树边和后向边的综合利用，使深度优先搜索可判断环的存在性



谢谢

BDA
Beihang University