Design and Analysis of Algorithms

Part IV: Graph Algorithms

Lecture 22: Review of Depth-First Search

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)

提纲

问题回顾

算法思想

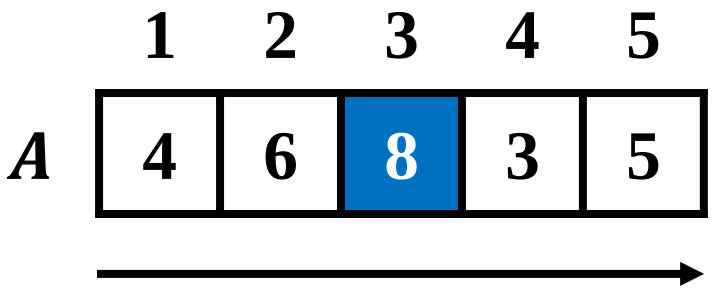
算法实例

算法分析

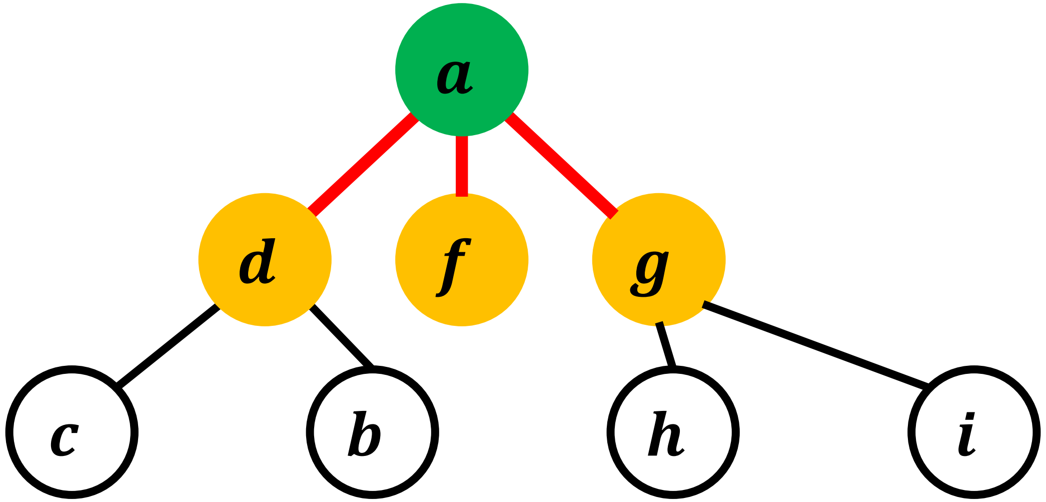
算法性质

图的搜索

- 数组结构
  - 查询最大值：简单循环搜索所有元素，记录最大值



- 图结构
  - 查询相邻顶点：简单循环搜索各顶点关联的边
  - 查询可达顶点：简单循环搜索，不能找到全部可达顶点！是否存在有效算法?



按照什么次序搜索顶点?

广度优先搜索

深度优先搜索

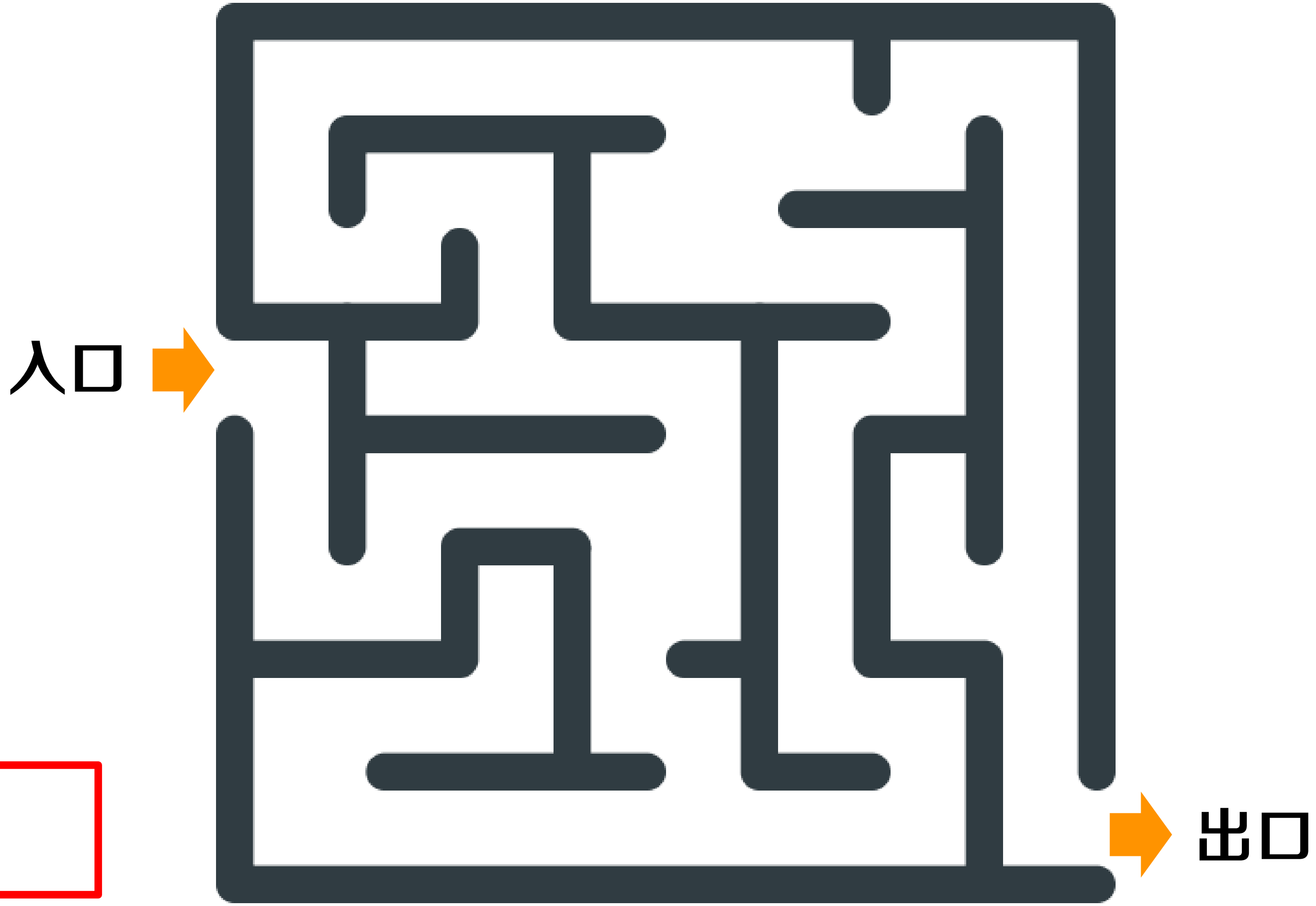
提纲

- 问题回顾
- **算法思想**
- 算法实例
- 算法分析
- 算法性质

算法思想

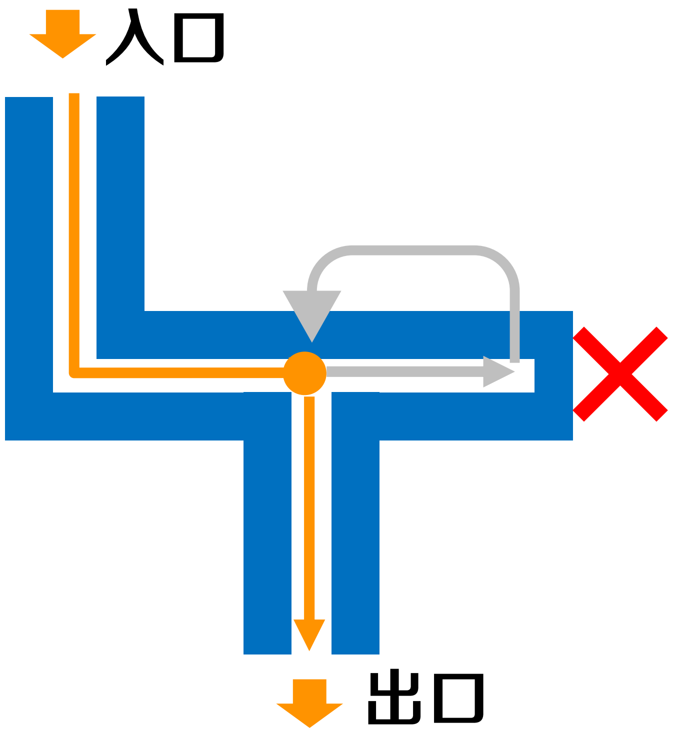
- 走迷宫

问题：如何走出迷宫?



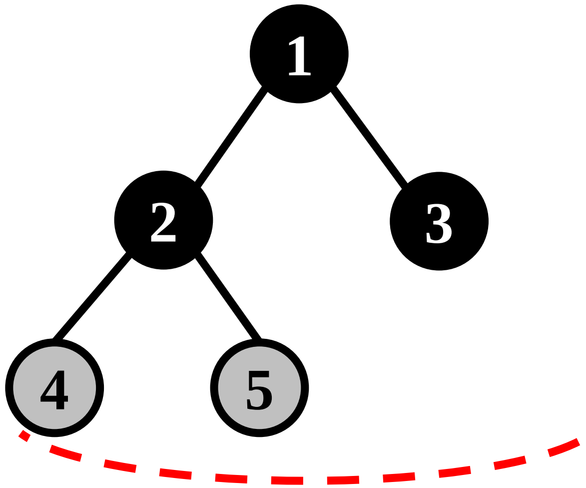
算法思想

- 算法步骤
  - 分叉时，任选一条边深入
  - 无边时，后退一步找新边
  - 找到边，从新边继续深入



算法思想

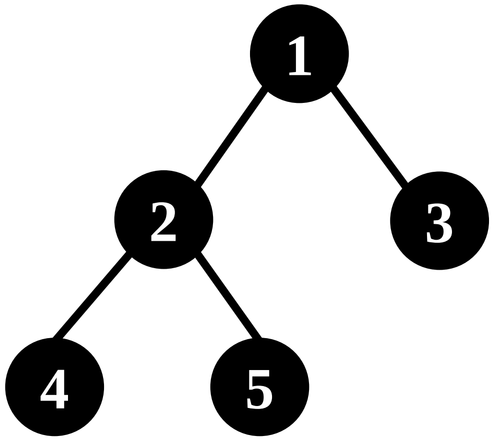
- 广度优先搜索



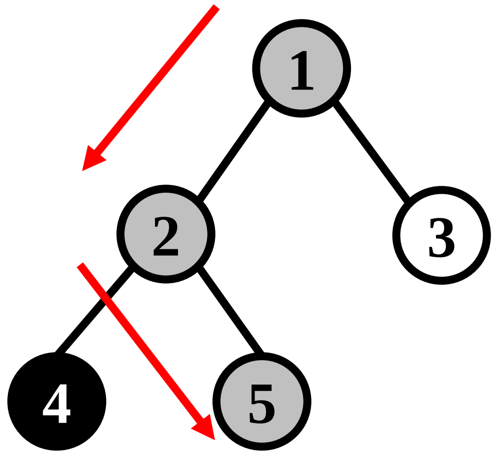
- 深度优先搜索

算法思想

- 广度优先搜索

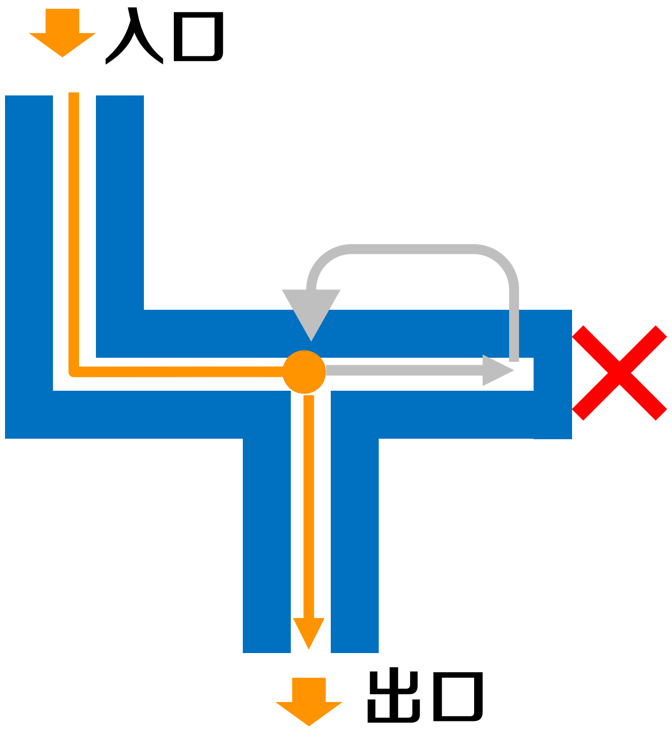


- 深度优先搜索



算法思想

- 算法步骤
  - 分叉时，任选一条边深入
  - 无边时，后退一步找新边
  - 找到边，从新边继续深入



- 辅助数组
  - $color$：用颜色表示顶点状态
    - $White$：白色顶点尚未被发现
    - $Black$：黑色顶点已被处理
    - $Gray$：正在处理，尚未完成
  - $pred$：顶点u由$pred[u]$发现
  - d：顶点发现时刻（变成灰色的时刻）
  - f：顶点完成时刻（变成黑色的时刻）

提纲

问题回顾

算法思想

算法实例

算法分析

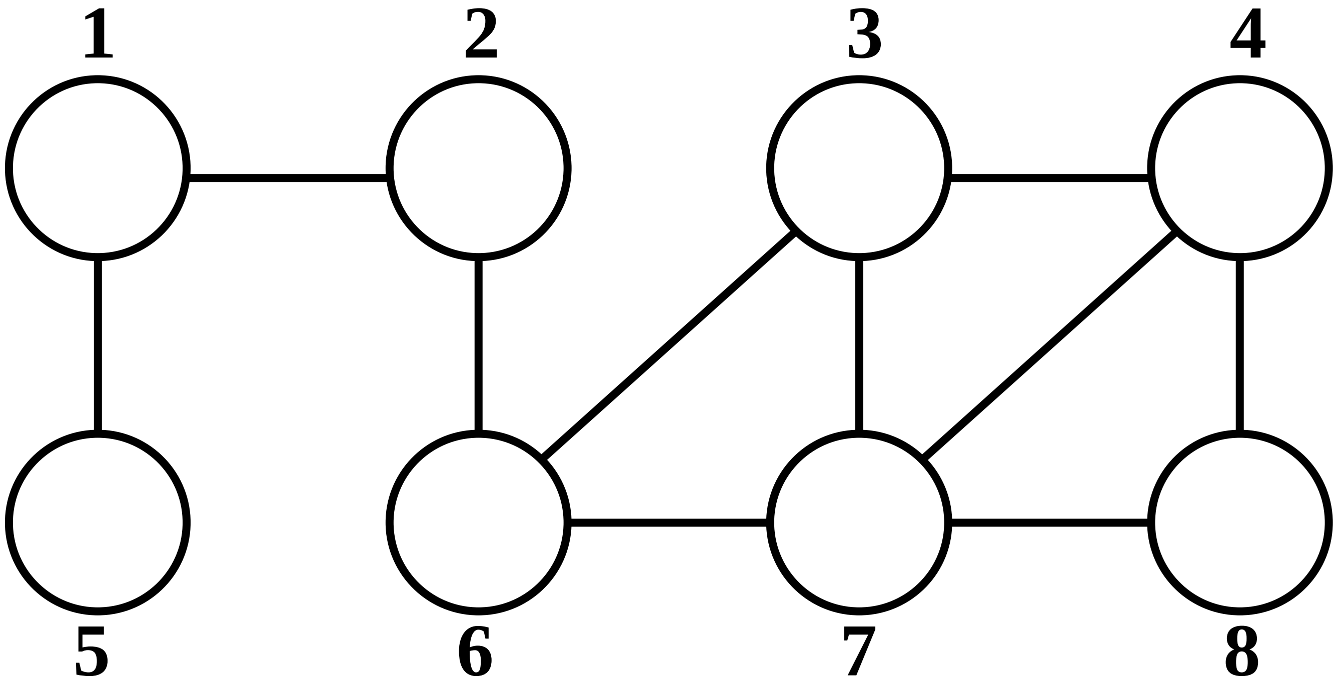
算法性质

算法实例

V	1	2	3	4	5	6	7	8
$color$	W	W	W	W	W	W	W	W
$pred$	N	N	N	N	N	N	N	N
d								
f								

$time = 0$

搜索源点

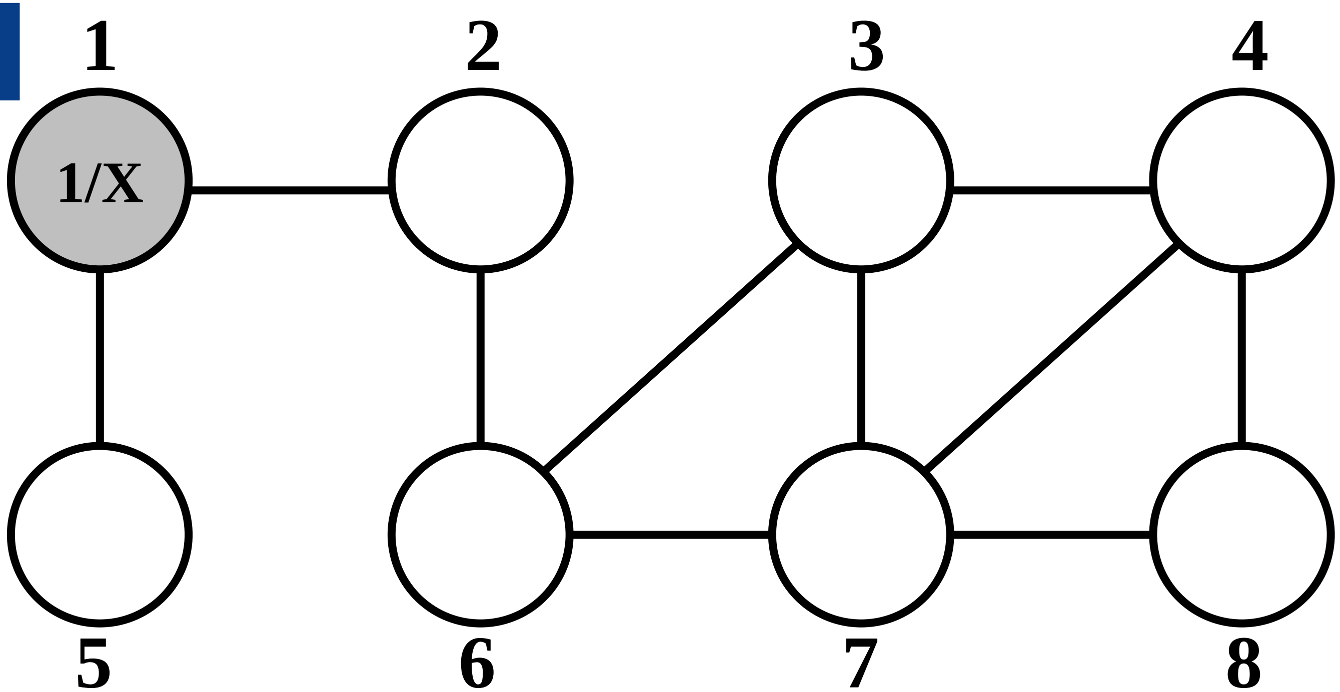


算法实例

V	1	2	3	4	5	6	7	8
color	G	W	W	W	W	W	W	W
pred	N	N	N	N	N	N	N	N
d	1							
f								

$time = 1$

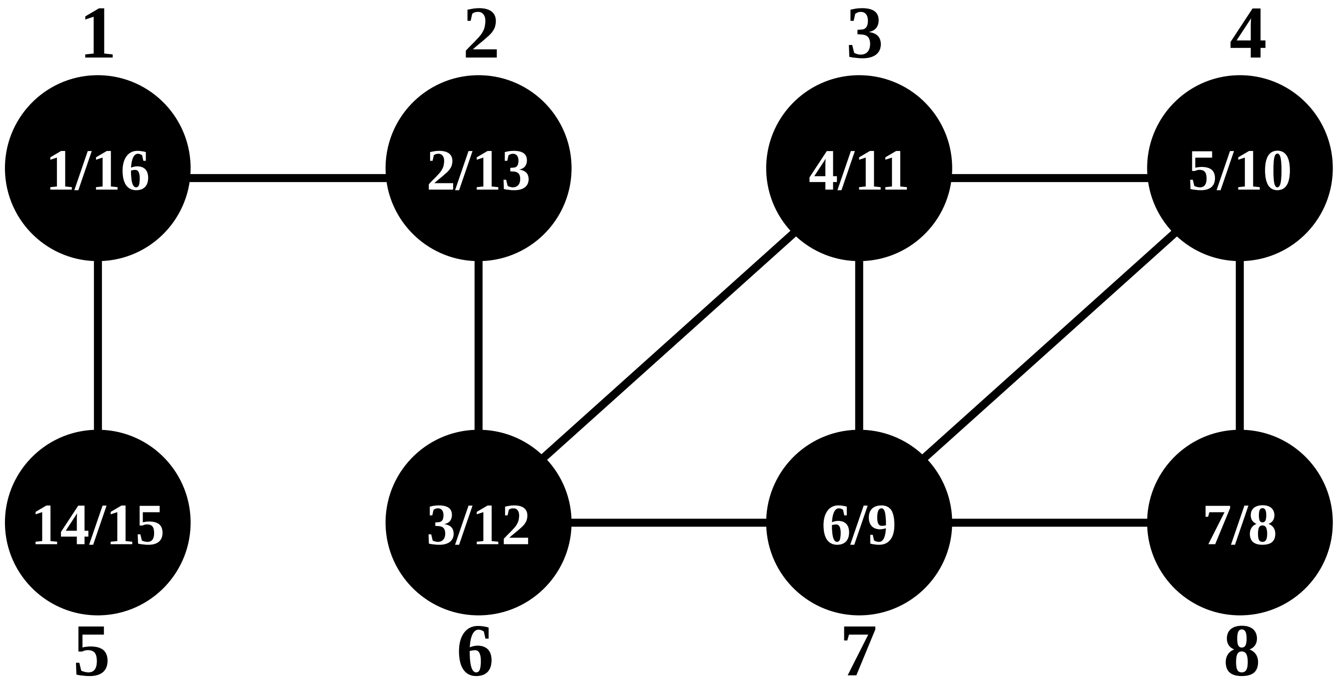
发现时刻为1



算法实例

V	1	2	3	4	5	6	7	8
color	B	B	B	B	B	B	B	B
pred	N	1	6	3	1	2	4	7
d	1	2	4	5	14	3	6	7
f	16	13	11	10	15	12	9	8

$time = 16$



提纲

问题回顾

算法思想

算法实例

算法分析

算法性质

伪代码

- **DFS(G)**

输入: 图G
输出: 祖先数组$pred$，发现时刻d，结束时刻f
新建数组 $color[1..V], pred[1..V], d[1..V], f[1..V]$
//初始化
for $v \in V$ **do**
    $pred[v] \leftarrow NULL$
    $color[v] \leftarrow WHITE$
end
$time \leftarrow 0$
for $v \in V$ **do**
    if $color[v] = WHITE$ **then**
        DFS-Visit(G, v)
    end
end
return $pred, d, f$

伪代码

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

```
输入: 图G, 顶点v
color[v] ← GRAY
time ← time + 1
d[v] ← time
for w ∈ G.Adj[v] do
    if color[w] = WHITE then
        pred[w] ← v
        DFS-Visit(G, w)
    end
end
color[v] ← BLACK
time ← time + 1
f[v] ← time
```

复杂度分析

- 尝试递归算法常用的主定理和递归树分析

输入: 图G, 顶点v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ **do**

  if $color[w] = WHITE$ **then**

    $pred[w] \leftarrow v$

    DFS-Visit(G, w)

  end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

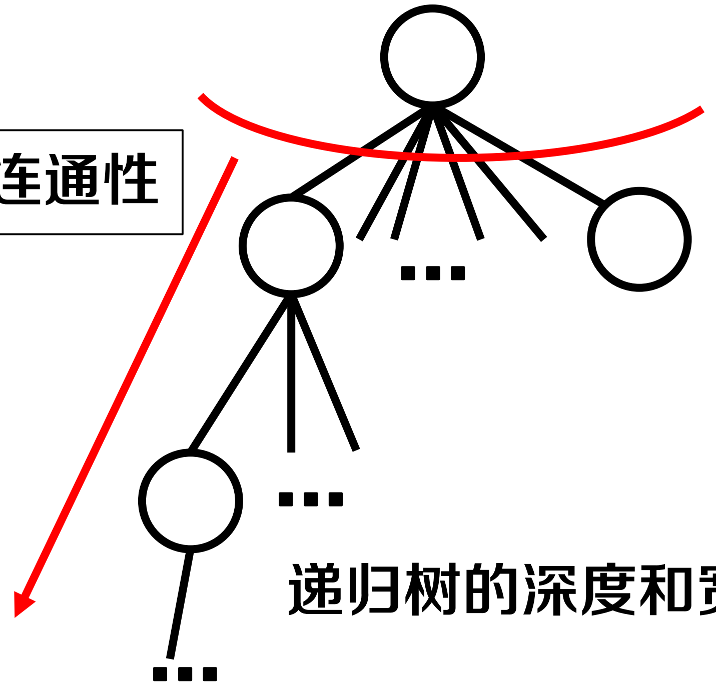
$f[v] \leftarrow time$

子问题规模取决于顶点的连通性

子问题个数：不确定

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

子问题规模：不确定



递归树的深度和宽度均不确定

能否借助广度优先搜索复杂度分析思想？

回顾：广度优先搜索算法复杂度分析

- 对于每个顶点u，搜索相邻顶点消耗时间$T_u = O(1 + deg(u))$
- 总运行时间：

分析每个顶点的时间开销

$$
\begin{aligned}
T &= \sum_{u \in V} T_u \\
&\le \sum_{u \in V} O(1 + deg(u)) \\
&= \sum_{u \in V} O(1) + \sum_{u \in V} O(deg(u)) \\
&= O(|V| + |E|)
\end{aligned}
$$

广度优先搜索的时间复杂度为$O(|V| + |E|)$

复杂度分析

输入: 图G, 顶点v

$color[v] \leftarrow GRAY$
$time \leftarrow time + 1$
$d[v] \leftarrow time$

$O(1)$

for $w \in G.Adj[v]$ **do**
  if $color[w] = WHITE$ **then**
    $pred[w] \leftarrow v$
    DFS-Visit(G, w)
  end
end

搜索顶点w的开销

$O(|Adj[v]|)$

$color[v] \leftarrow BLACK$
$time \leftarrow time + 1$
$f[v] \leftarrow time$

$O(1)$

搜索顶点v的开销
$O(1 + |Adj[v]|)$

- 搜索顶点v的开销：$O(1 + |Adj[v]|)$

复杂度分析

输入: 图G, 顶点v

$color[v] \leftarrow GRAY$
$time \leftarrow time + 1$
$d[v] \leftarrow time$
for $w \in G.Adj[v]$ **do**
  if $color[w] = WHITE$ **then**
    $pred[w] \leftarrow v$
    DFS-Visit(G, w)
  end
end
$color[v] \leftarrow BLACK$
$time \leftarrow time + 1$
$f[v] \leftarrow time$

搜索后不是白色

只有白色才搜索

搜索后不是白色

- 搜索顶点v的开销：$O(1 + |Adj[v]|)$
- 每个顶点只搜索一次，共$|V|$次

复杂度分析

输入: 图G, 顶点v

$color[v] \leftarrow GRAY$
$time \leftarrow time + 1$
$d[v] \leftarrow time$
for $w \in G.Adj[v]$ **do**
  if $color[w] = WHITE$ **then**
    $pred[w] \leftarrow v$
    DFS-Visit(G, w)
  end
end
$color[v] \leftarrow BLACK$
$time \leftarrow time + 1$
$f[v] \leftarrow time$

- **搜索顶点v的开销：$O(1 + |Adj[v]|)$**
- **每个顶点只搜索一次，共$|V|$次**
- **总时间复杂度**
  - $O(\sum_{v \in V}(1 + |Adj[v]|)) = O(|V| + \sum_{v \in V} |Adj[v]|) = O(|V| + |E|)$

$\sum_{v \in V} deg(v) = O(|E|)$

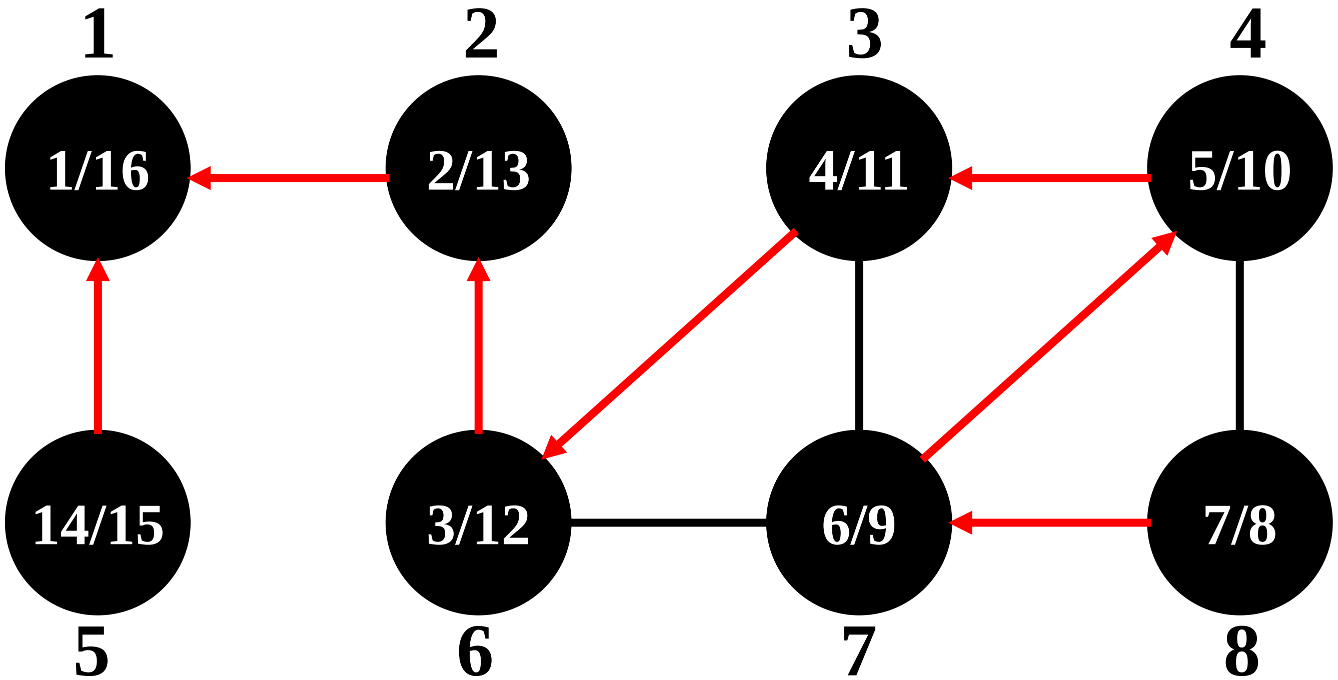
提纲

- 问题回顾
- 算法思想
- 算法实例
- 算法分析
- **算法性质**

深度优先树

V	1	2	3	4	5	6	7	8
color	B	B	B	B	B	B	B	B
pred	N	1	6	3	1	2	4	7
d	1	2	4	5	14	3	6	7
f	16	13	11	10	15	12	9	8

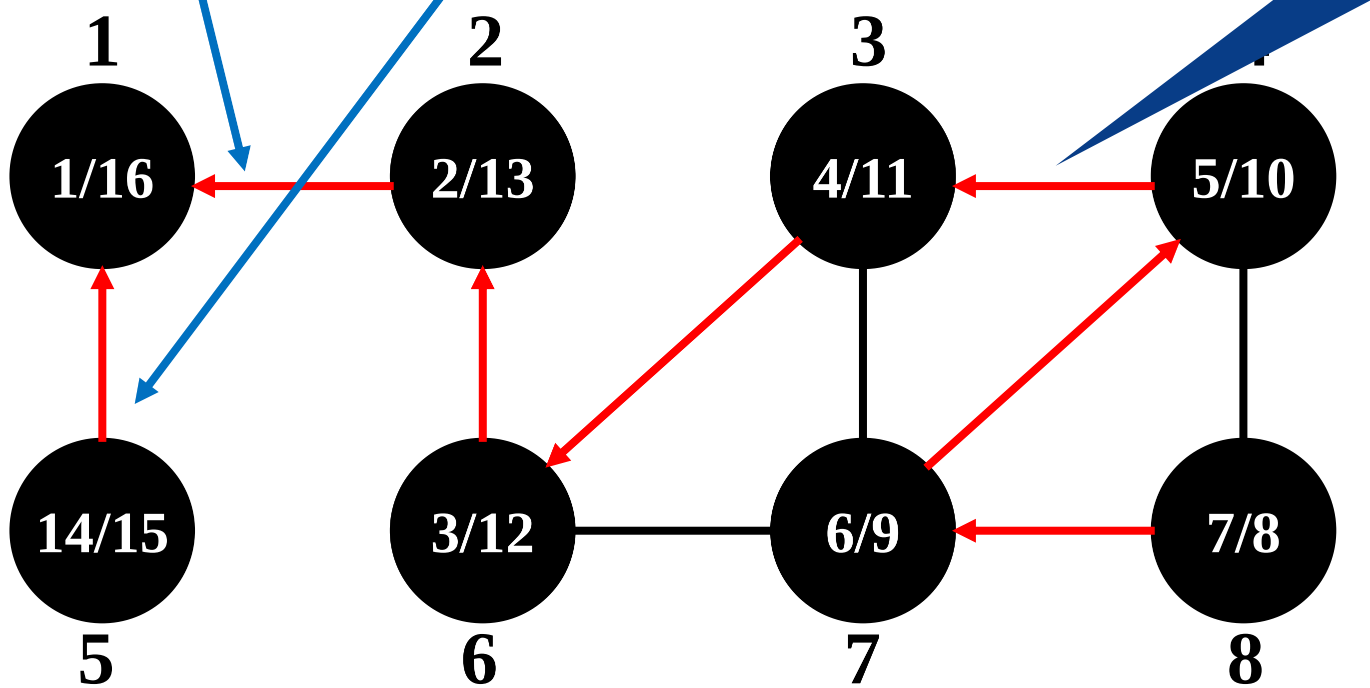
深度优先树：顶点以前驱为祖先形成的树



深度优先树

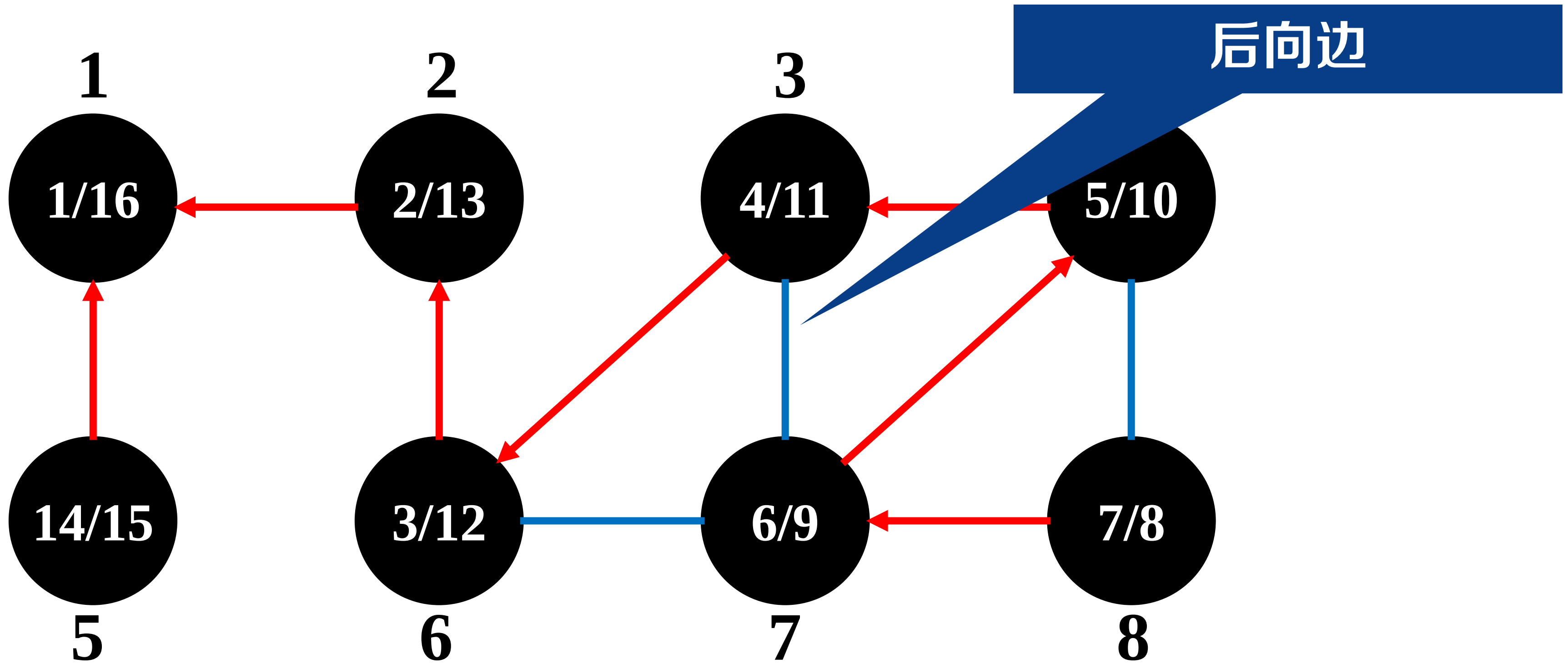
V	1	2	3	4	5	6	7	8
color	B	B	B	B	B	B	B	B
pred	N	1	6	3	1	2	4	7
d	1	2	4	5	14	3	6	7
f	16	13	11	10	15	12	9	8

树边：深度优先中的边



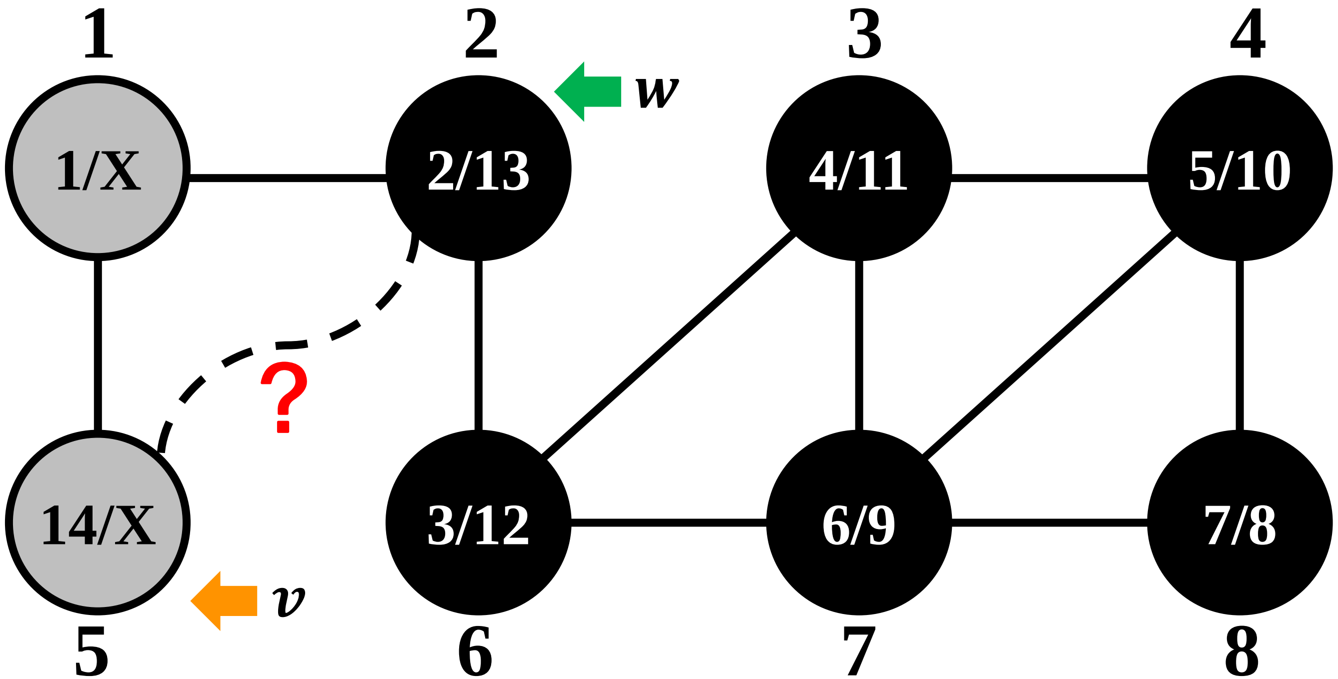
边的性质

- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系



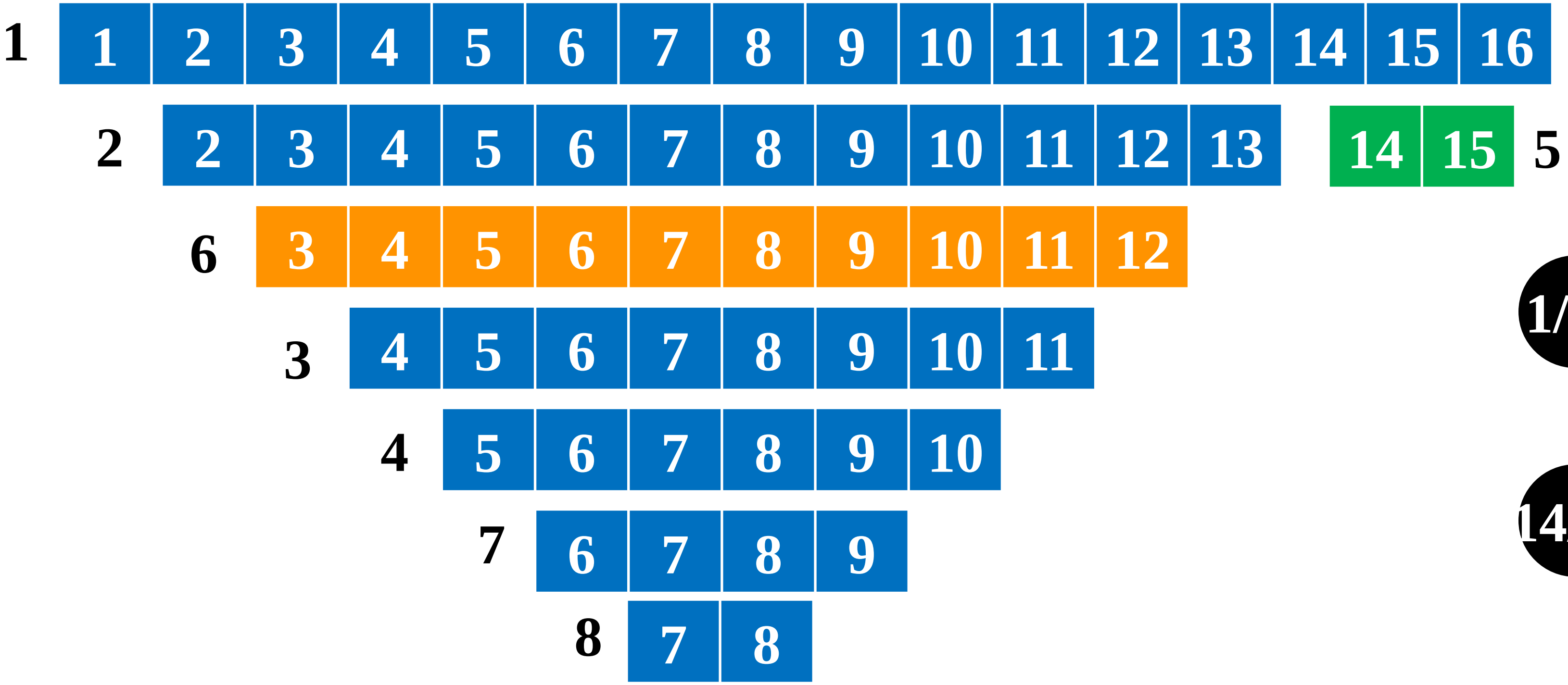
边的性质

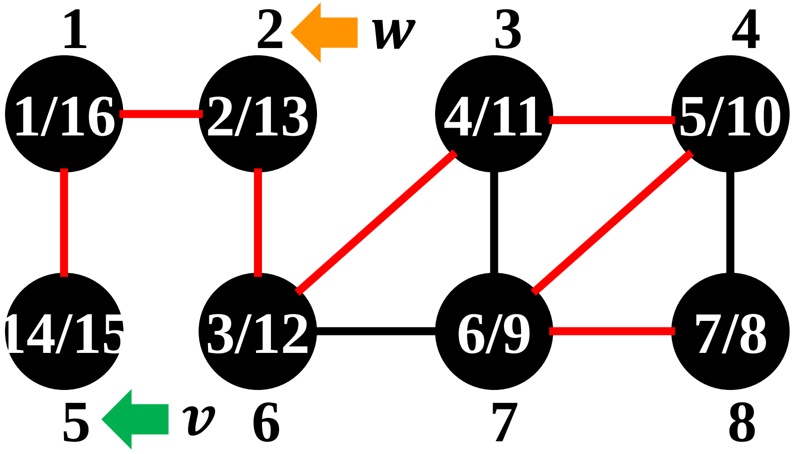
- 后向边：不是树边，但两顶点在深度优先树中是祖先后代关系
- 对于无向图，非树边一定是后向边
  - 证明：从顶点v，搜索顶点w
    - 若w为白色，(v, w)为树边
    - 若w为灰色，(v, w)为后向边
    - 若w为黑色？不可能！若存在图中虚线边，w变黑前一定已经搜索过v



点的性质

- 括号化定理
  - 点v发现时刻和结束时刻构成区间$[d[v], f[v]]$
  - 任意两点v, w必满足以下情况之一：
    - $[d[v], f[v]]$包含$[d[w], f[w]]$，w是v的后代
    - $[d[w], f[w]]$包含$[d[v], f[v]]$，v是w的后代
    - $[d[v], f[v]]$和$[d[w], f[w]]$完全不重合，v和w均不是对方后代





点的性质

- **括号化定理**
  - **证明**

观察1：仅在区间内为灰色

观察2：白点必是灰点后代

输入: 图G, 顶点v

$color[v] \leftarrow GRAY$

$time \leftarrow time + 1$

$d[v] \leftarrow time$

for $w \in G.Adj[v]$ **do**

  if $color[w] = WHITE$ **then**

    $pred[w] \leftarrow v$

    DFS-Visit(G, w)

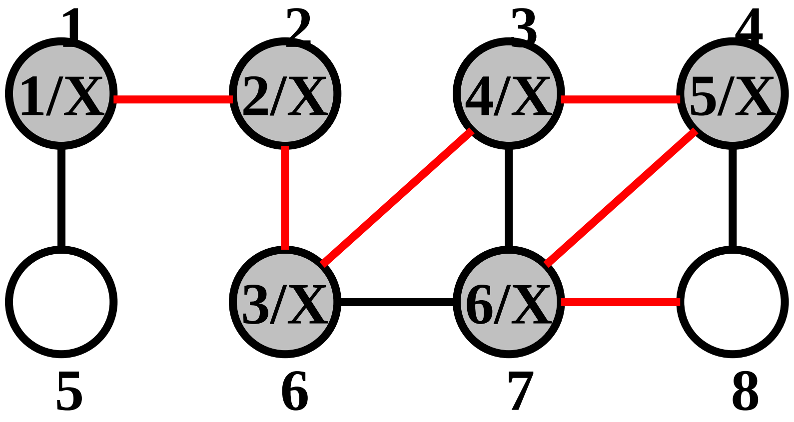
  end

end

$color[v] \leftarrow BLACK$

$time \leftarrow time + 1$

$f[v] \leftarrow time$



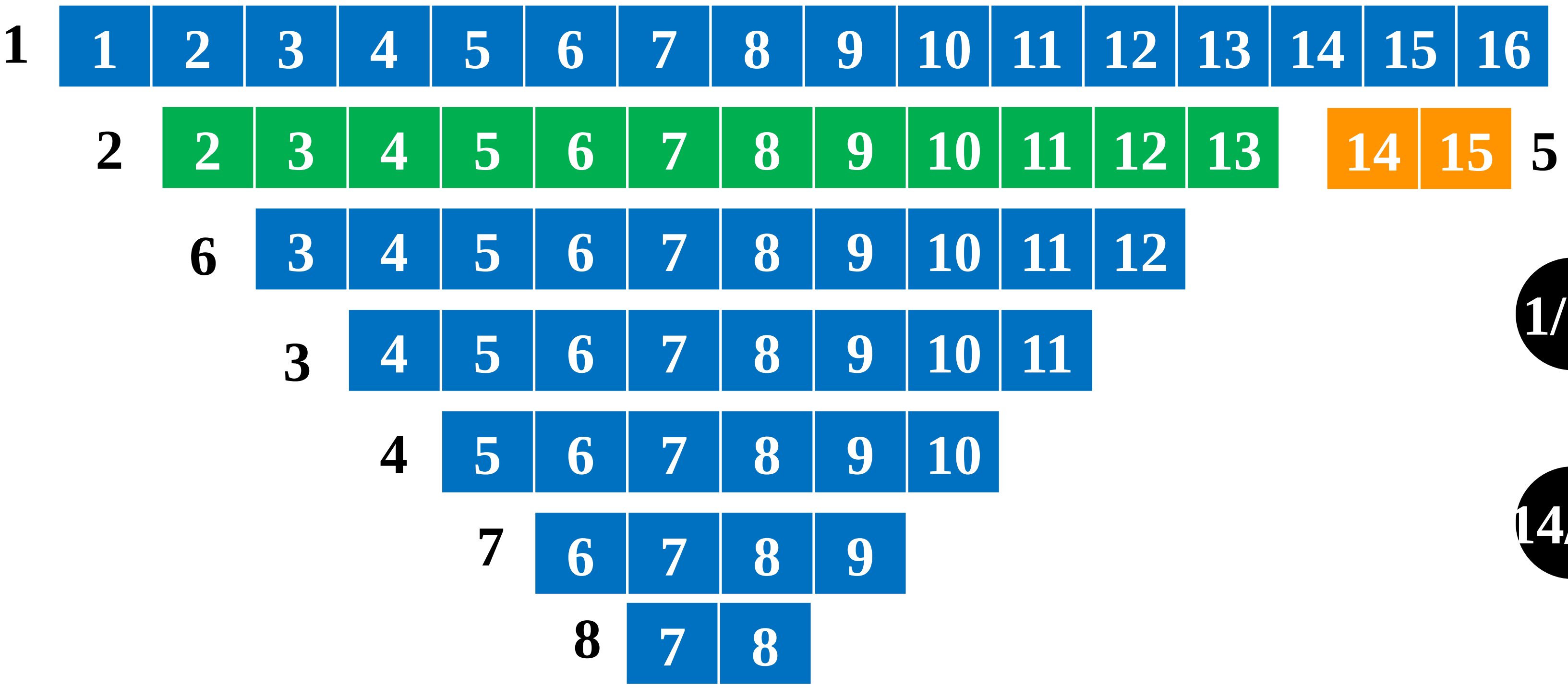
$d[v]$ $f[v]$

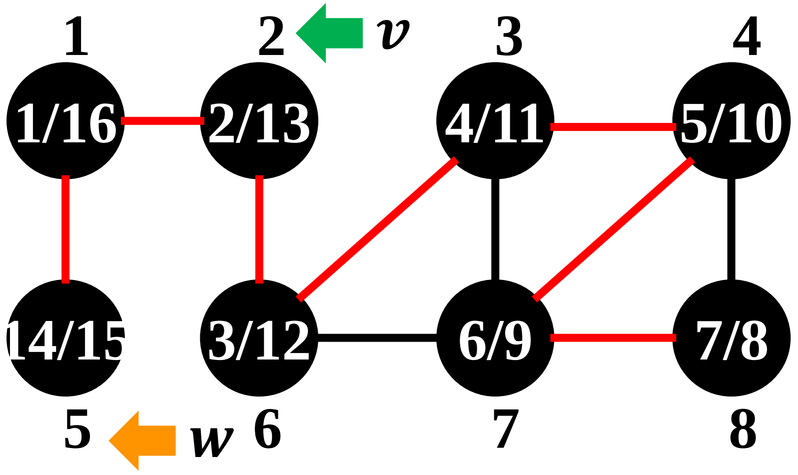
点的性质

- 括号化定理
  - 证明(不妨设$d[v] < d[w]$)

观察1：仅在区间内为灰色　　观察2：白点必是灰点后代

- 若$f[v] > d[w]$：w被发现时，v为灰色，所以w是v的后代
- 若$f[v] < d[w]$：w被发现时，v为黑色，所以v和w均不是对方后代



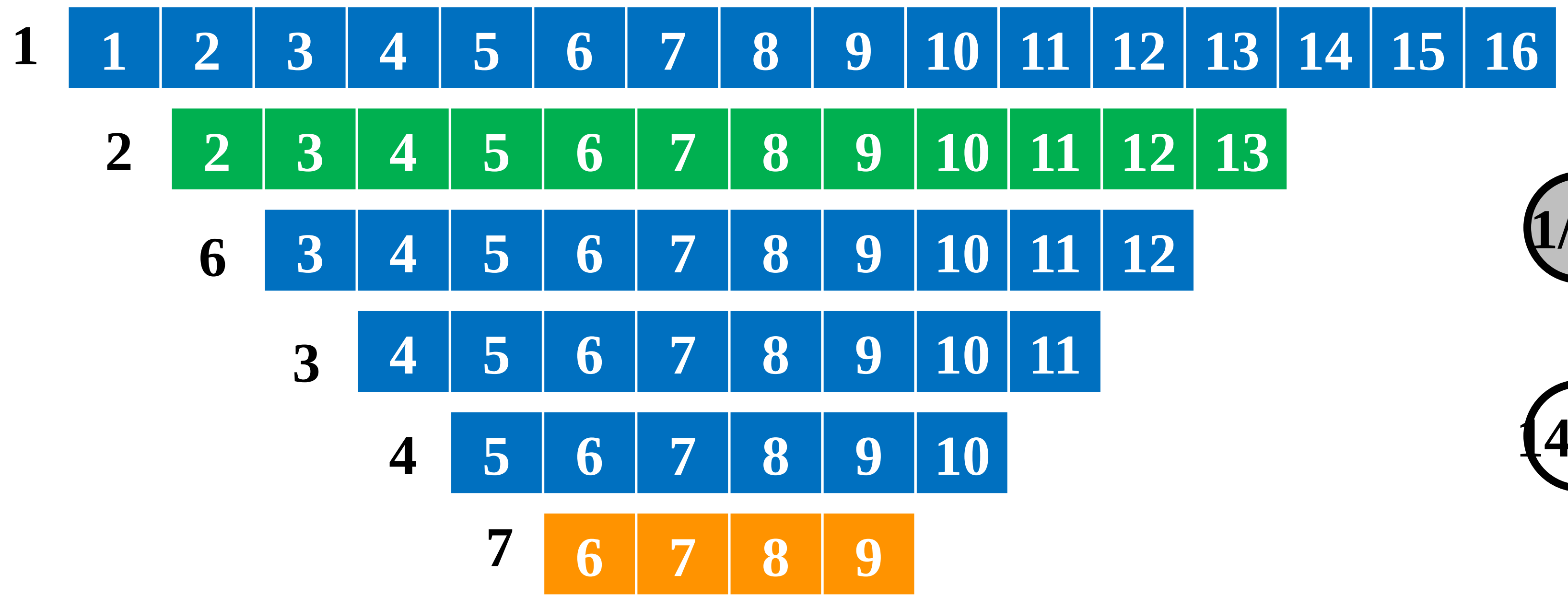


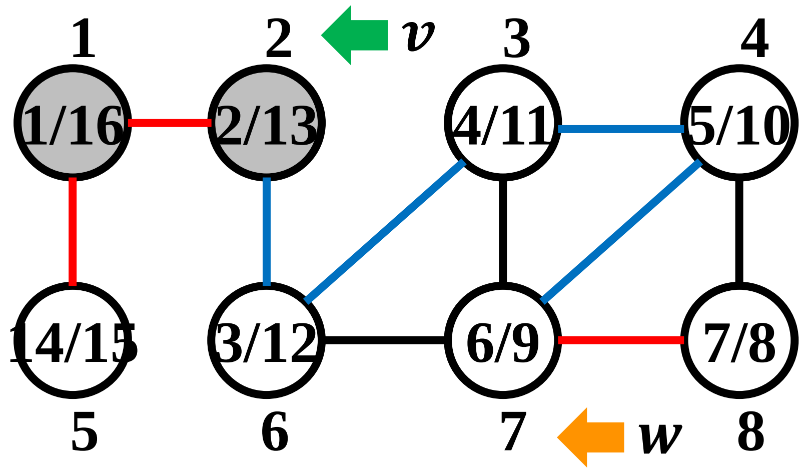
路径性质

- 白色路径定理
  - 在深度优先树中，顶点v是w的祖先$\Leftrightarrow$在v被发现前，从v到w存在全为白色顶点构成的路径
  - 证明（基本思想）

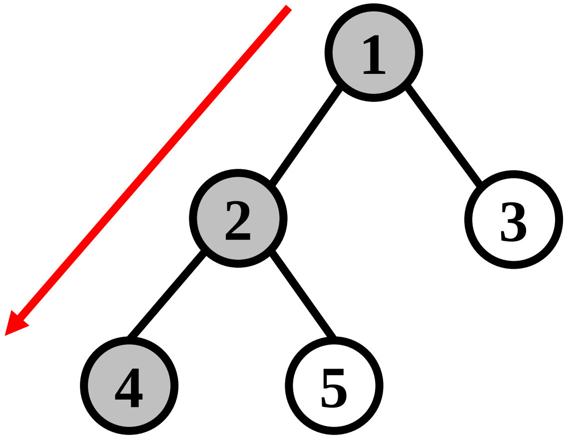
    $\Rightarrow$：由括号化定理，v在发现之前，其后代全为白色

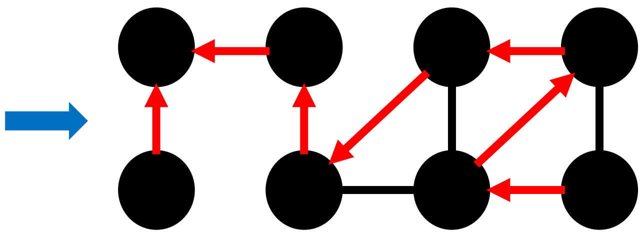
    $\Leftarrow$：v刚发现时，路径上除v以外的点仍都为白色，按深度优先搜索特点，一定会搜索路径上的所有点

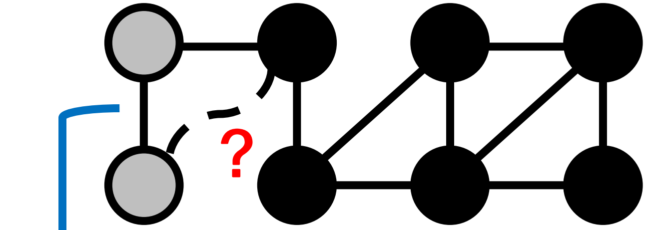




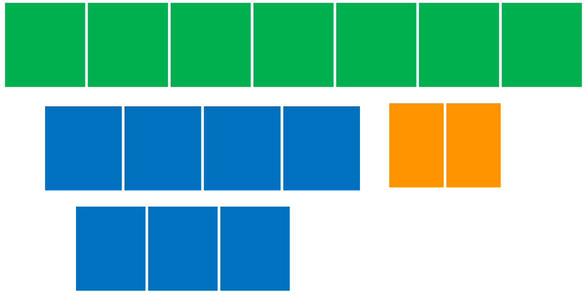
小结



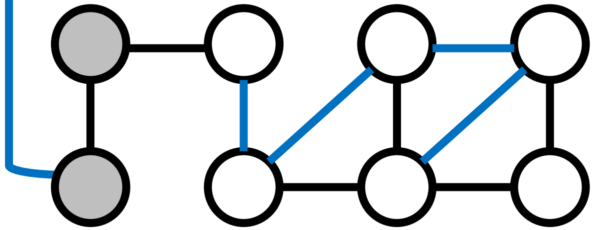




边的性质
无向图非树边为后向边



点的性质
括号化定理

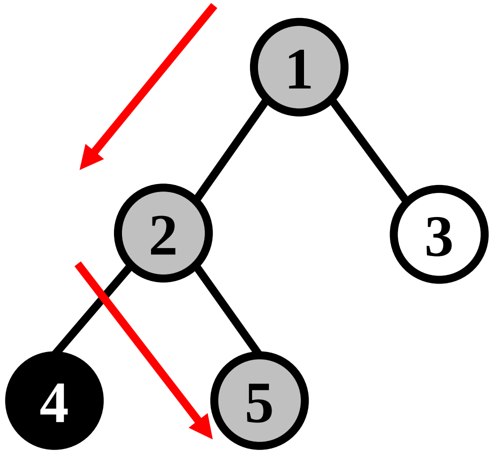


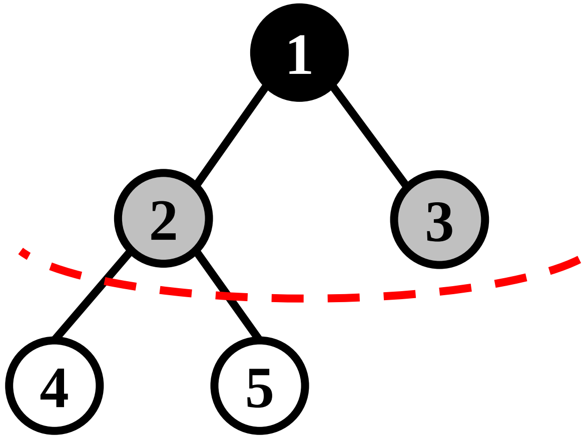
路径性质
白色路径定理

算法思想：深度优先

运行结果：深度优先树

小结

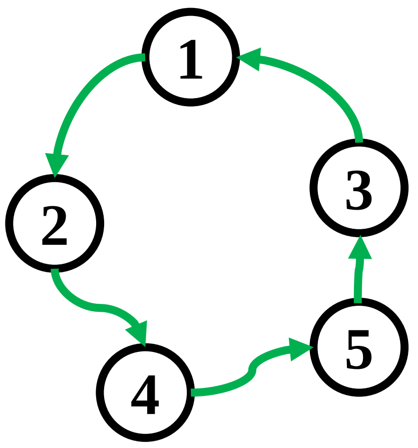


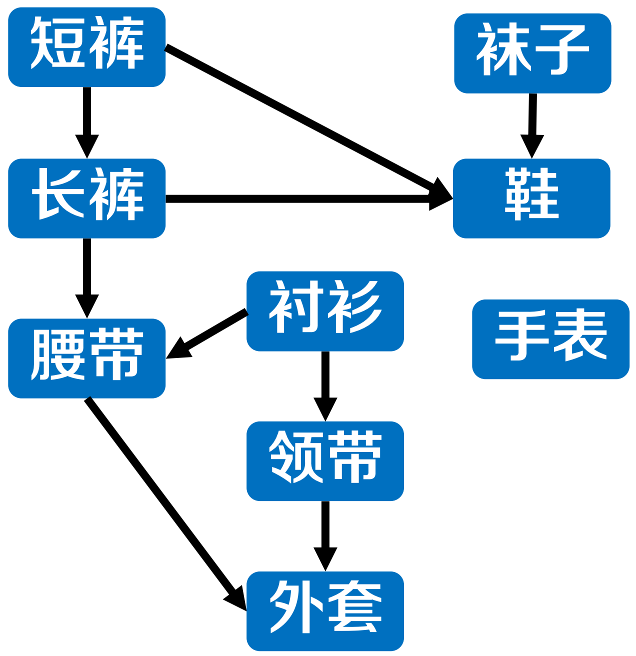


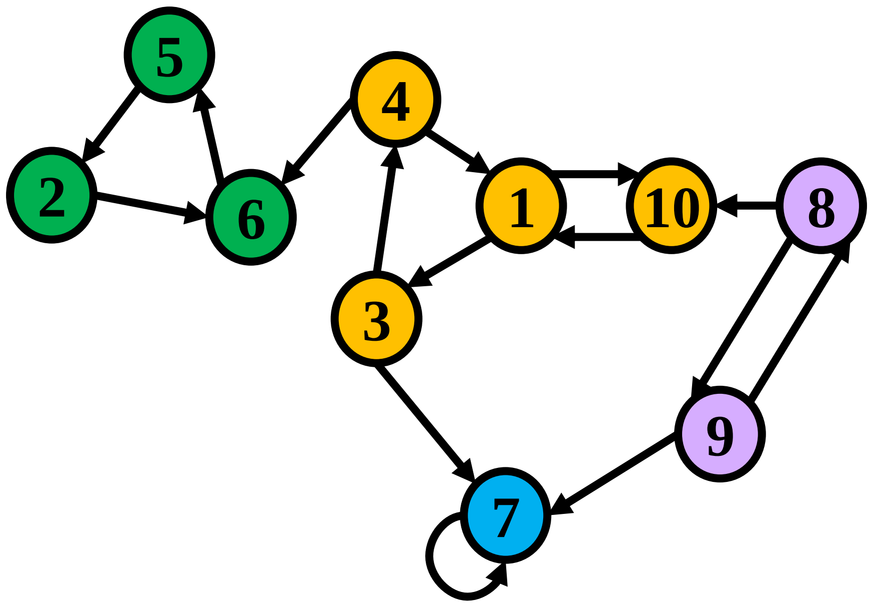
深度优先搜索：勇往直前

广度优先搜索：步步为营

算法应用







环路的存在性判断

拓扑排序

强连通分量

谢谢

BDA

Beihang University