极客时间算法训练营
第五课
树与图

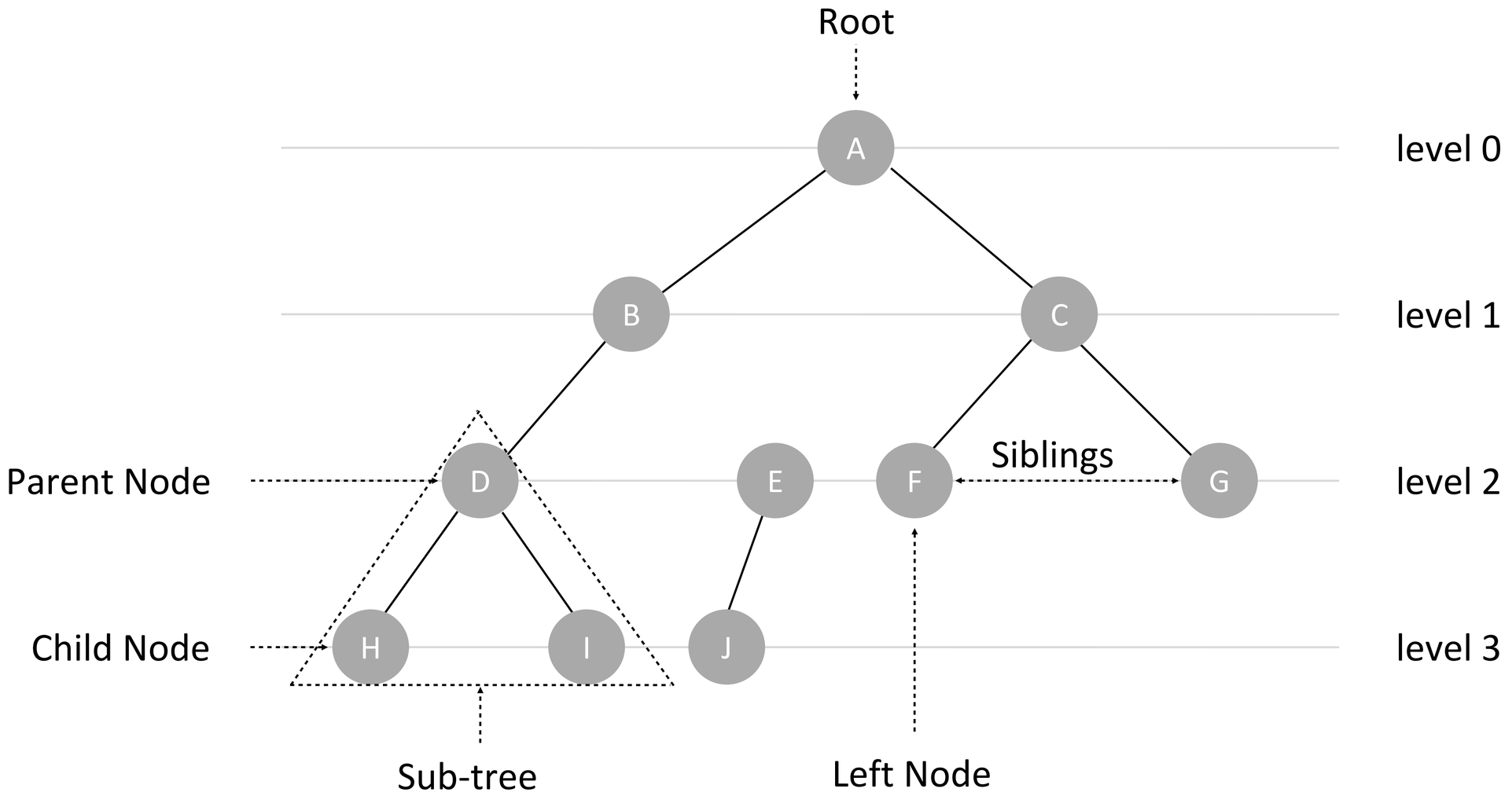
李煜东

《算法竞赛进阶指南》作者

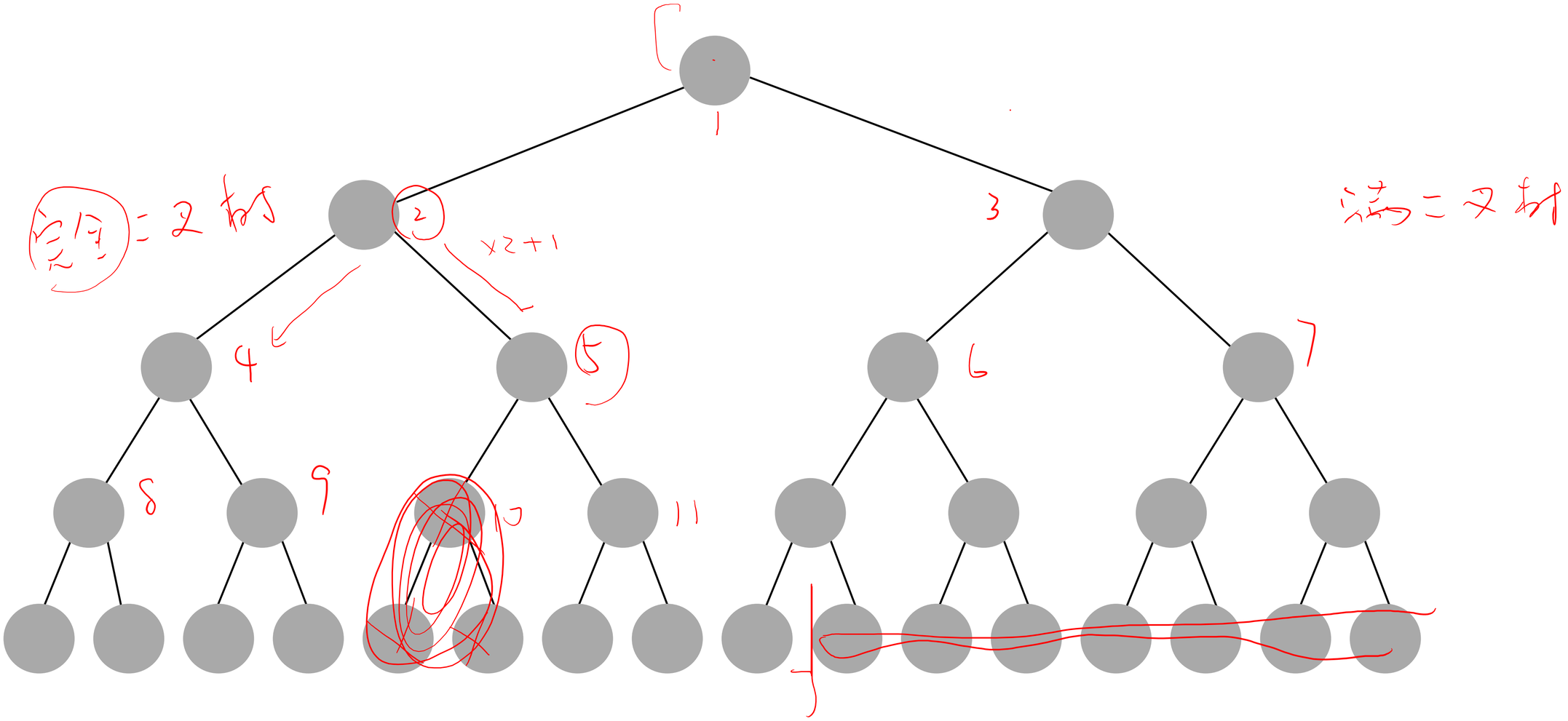
目录

树、二叉树、树的遍历

树 Tree


Root
A
B
C
D
E
F
G
H
I
J
level 0
level 1
level 2
level 3
Parent Node
Child Node
Siblings
Sub-tree
Left Node

二叉树 Binary Tree



代码 - 定义树的结点

C++

```cpp
struct TreeNode {
  int val;
  TreeNode *left;
  TreeNode *right;
  TreeNode(int x)
    : val(x), left(nullptr), right(nullptr) {}
}
```

Java

```java
public class TreeNode {
  public int val;
  public TreeNode left, right;
  public TreeNode(int val) {
      this.val = val;
      this.left = null;
      this.right = null;
  }
}
```

Python

```python
class TreeNode:
   def __init__(self, val):
     self.val = val
     self.left, self.right = None, None
```

二叉树的遍历

前序 Pre-order：根 - 左子树 - 右子树

A B D H I E J C F G

中序 In-order：左子树 - 根 - 右子树

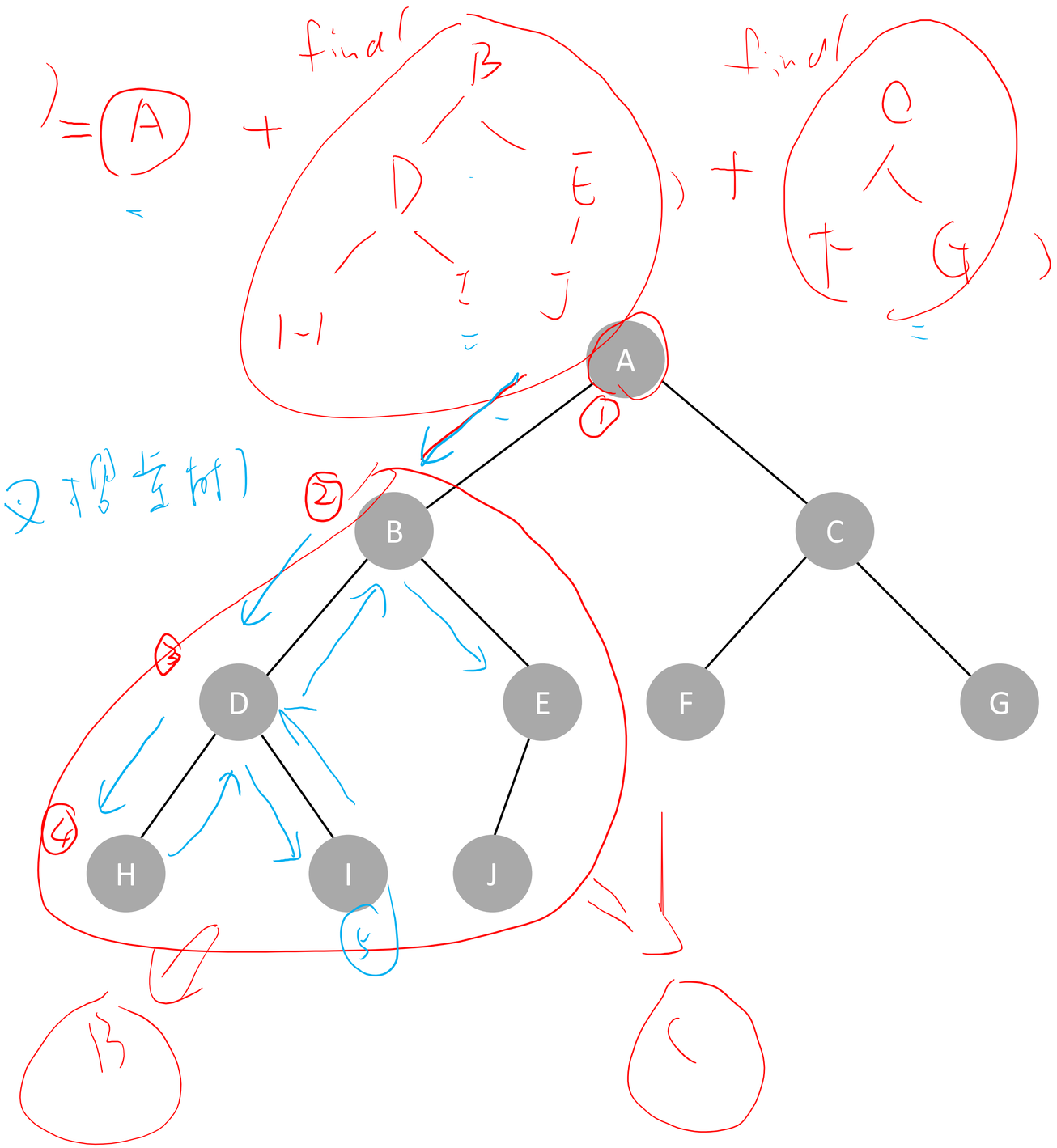
H D I B J E A F C G

后序 Post-order：左子树 - 右子树 - 根

H I D J E B F G C A

层次序

A B C D E F G H I J



树的遍历

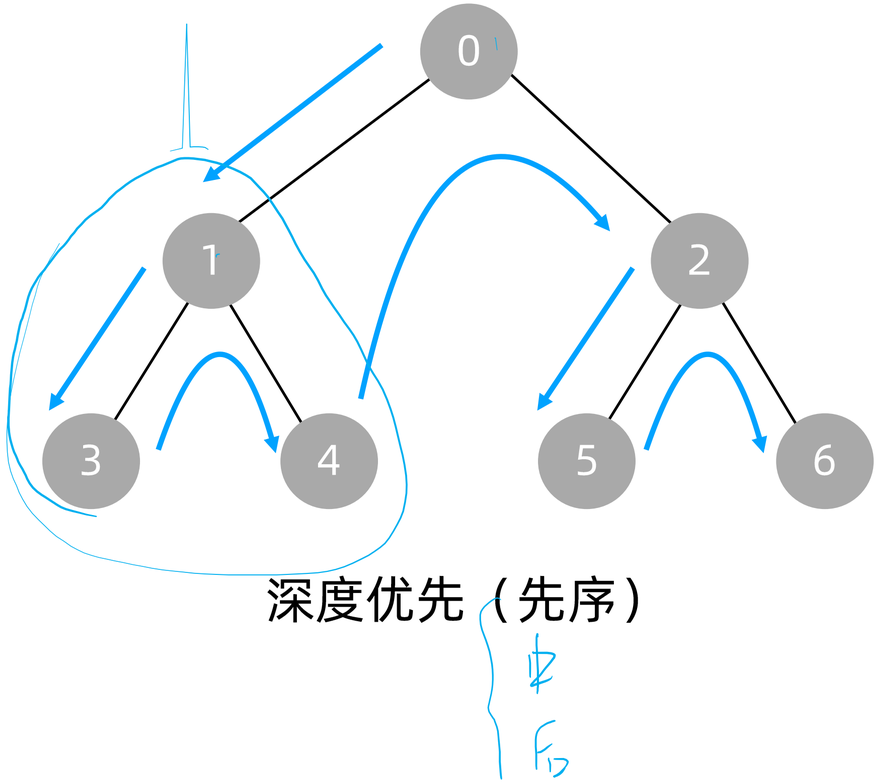
先序、中序、后序一般用递归来求

树的先序遍历又称树的深度优先遍历

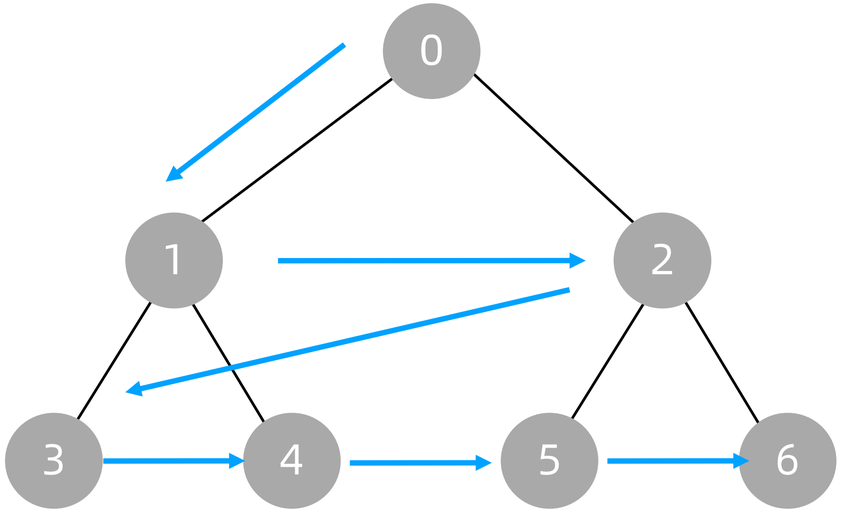
层次序一般借助队列来求

树的层序遍历又称树的广度优先遍历

树的遍历

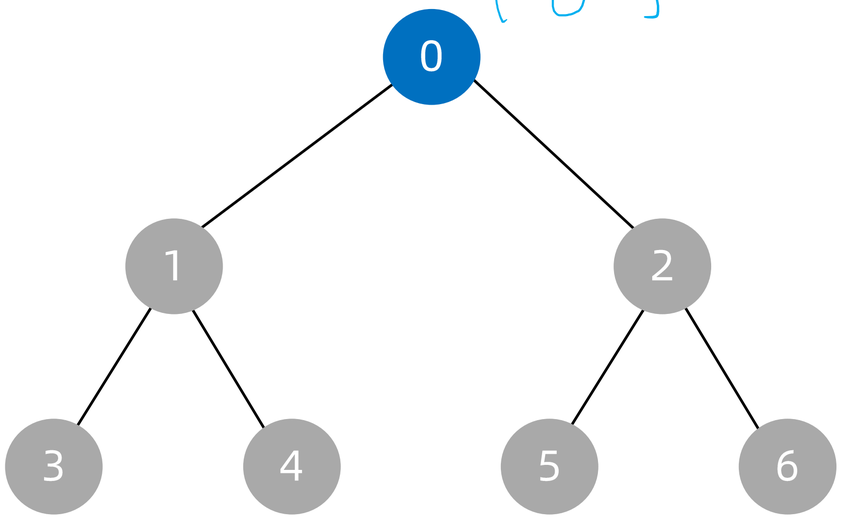


深度优先（先序）

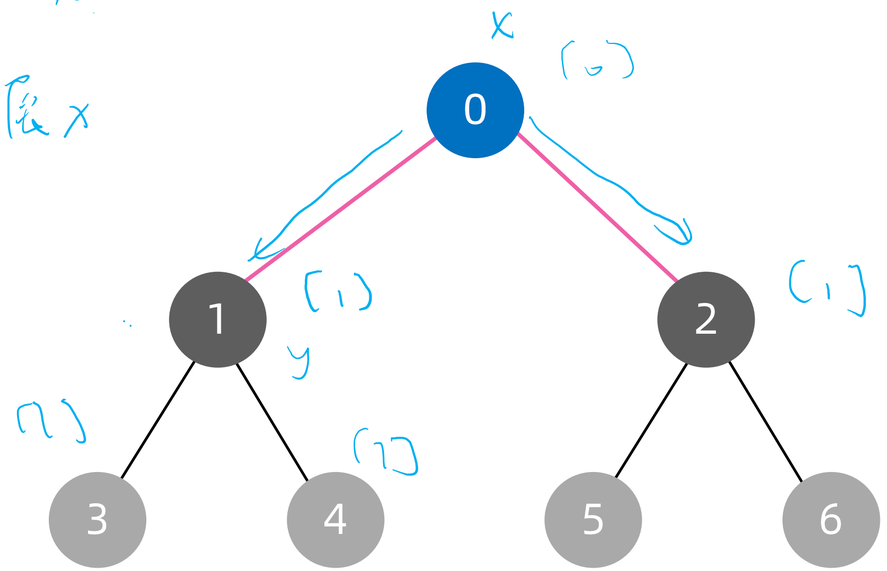


广度优先（层序）

广度优先遍历

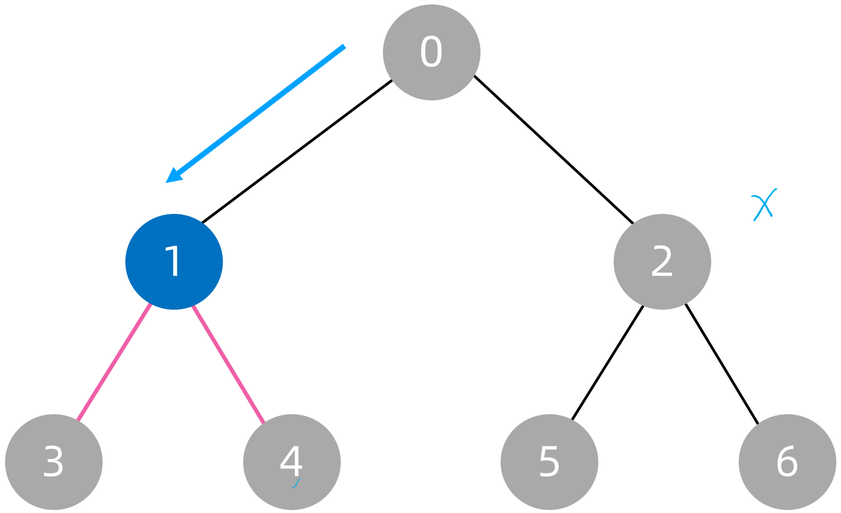


队列 0

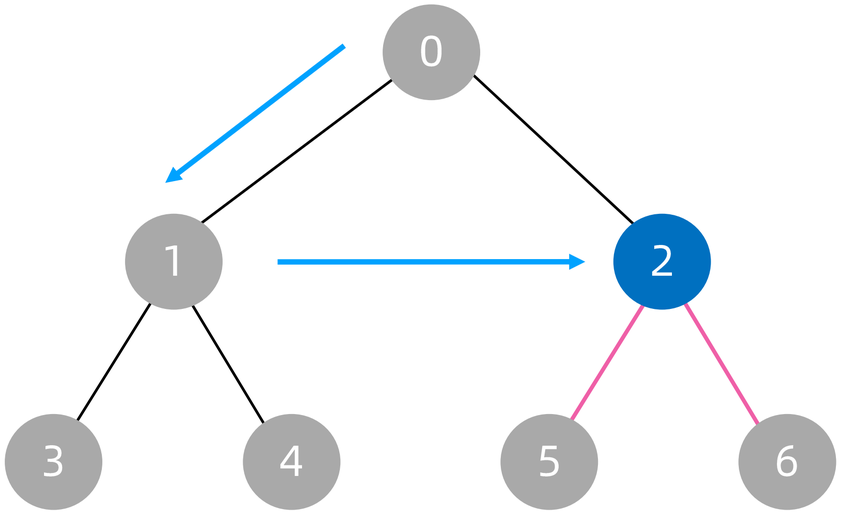


队列 2 3 4

广度优先遍历

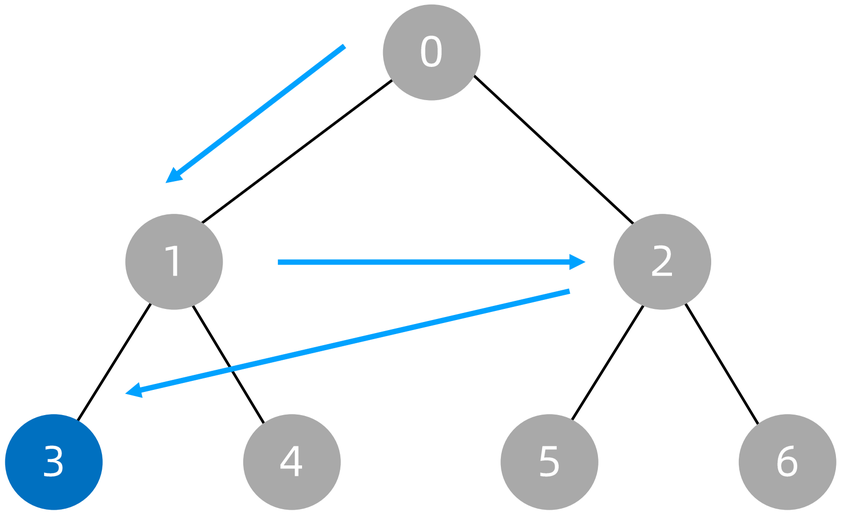


队列 2 3 4 5 6

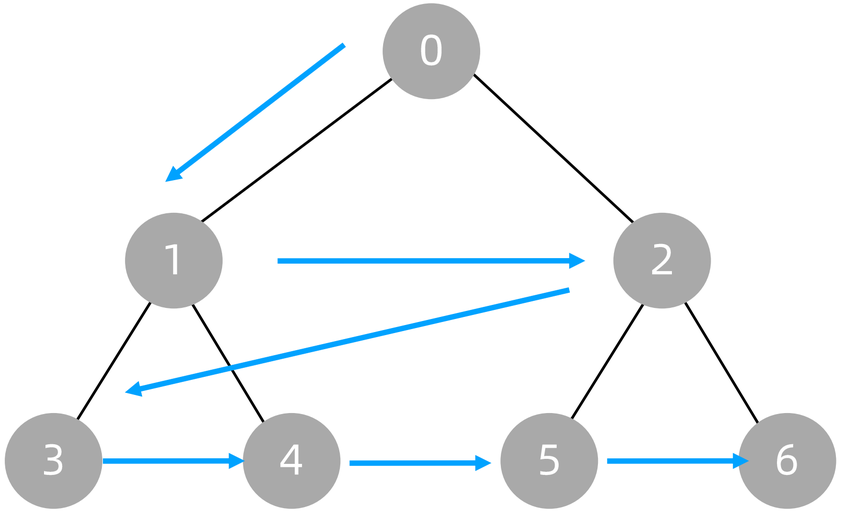


队列 3 4 5 6

广度优先遍历



队列　4 5 6



队列

实战

二叉树的中序遍历

https://leetcode-cn.com/problems/binary-tree-inorder-traversal/

N 叉树的前序遍历

https://leetcode-cn.com/problems/n-ary-tree-preorder-traversal/description/

N 叉树的层序遍历

https://leetcode-cn.com/problems/n-ary-tree-level-order-traversal/

二叉树的序列化与反序列化

https://leetcode-cn.com/problems/serialize-and-deserialize-binary-tree/

实战

从前序与中序遍历序列构造二叉树

https://leetcode-cn.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/

从中序与后序遍历序列构造二叉树（Homework）

https://leetcode-cn.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/

注意：从前序与后序遍历序列，并不能唯一确定一棵二叉树

树的直径、最近公共祖先、树的变形

树的直径（选做）

树的直径

https://leetcode-cn.com/problems/tree-diameter/

两次深度优先遍历

- 第一次从任意一个点出发，找到距离它最远的点 p
- 第二次从 p 出发，找到距离它最远的点 q
- 连接 p, q 的路径即为树的直径

最近公共祖先（LCA）

二叉树的最近公共祖先

https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-tree/

先求出父结点，然后用向上标记法

- p 向上一直到 root 标红色
- q 向上，第一次遇到红色时，就找到了LCA

基环树

向一棵树添加一条边，就形成了一个环

此时整个结构被称为基环树（pseudotree / unicyclic graph）

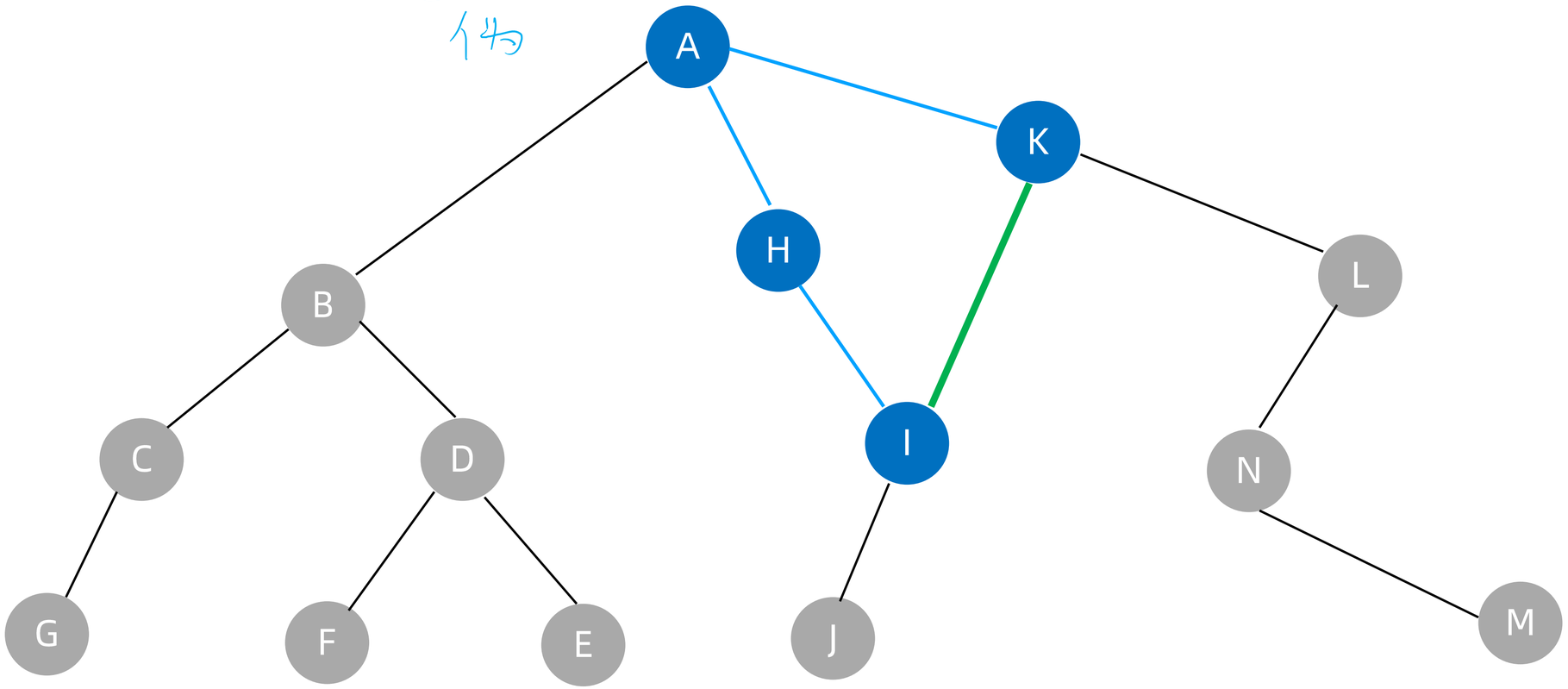
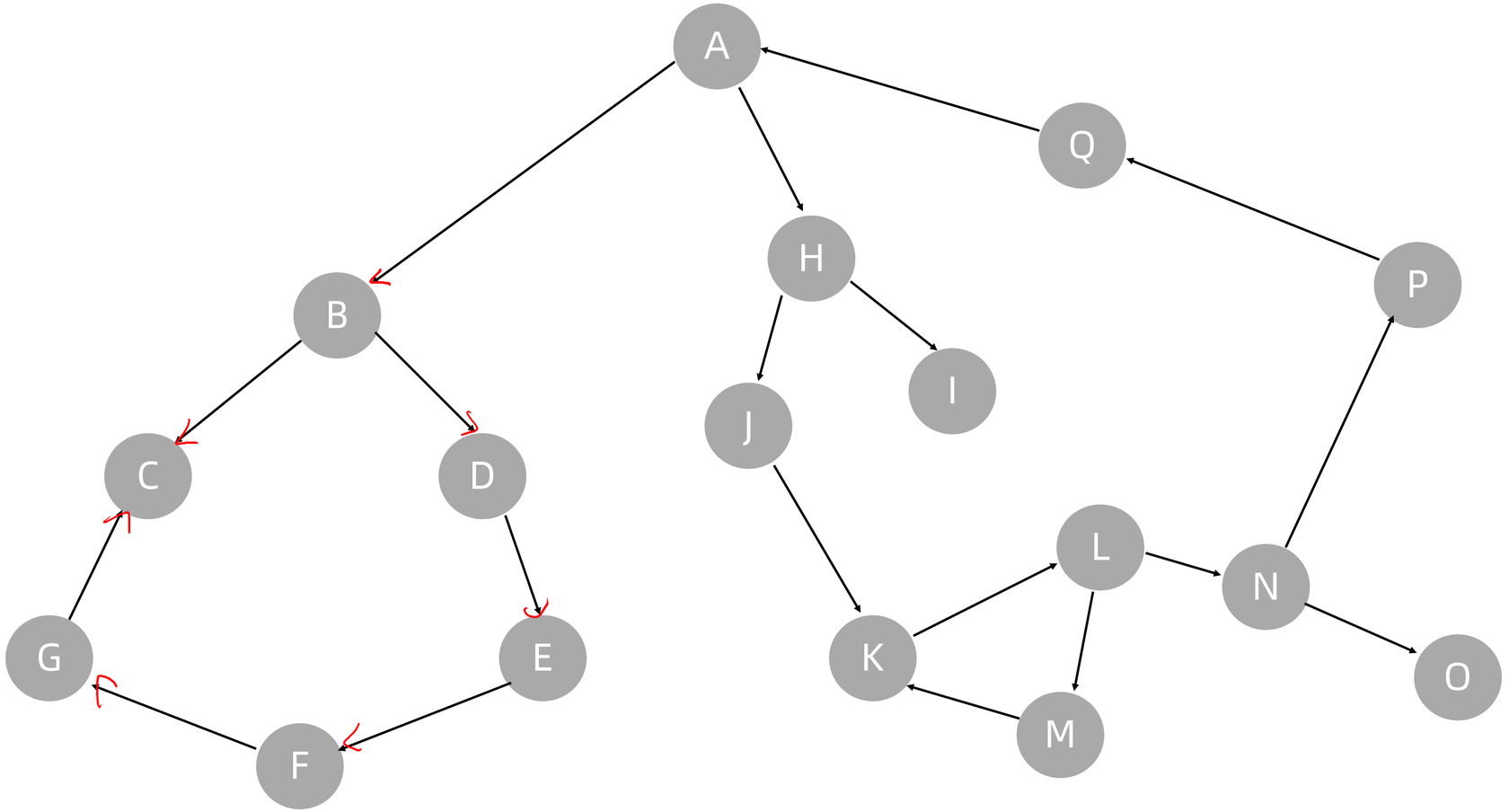


图 Graph



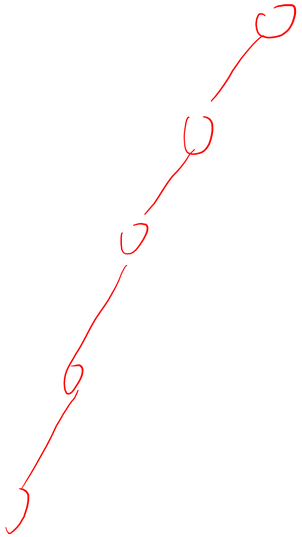
图、图的遍历

链表、树、图的关系

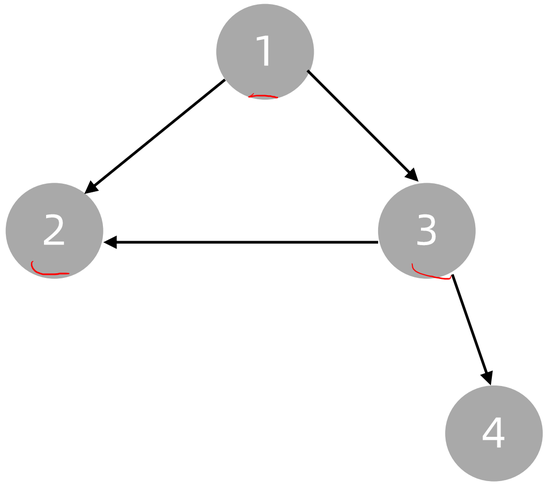
链表是特殊化的树

树是特殊化的图

- N 个点 N - 1 条边的连通无向图——树
- N 个点 N 条边的连通无向图——基环树



图的存储

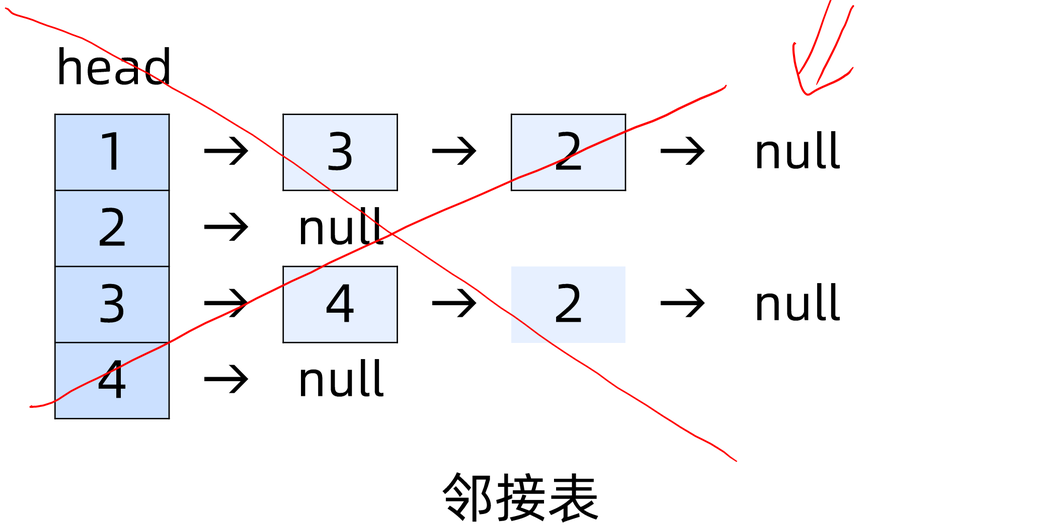


	1	2	3	4
1	0	1	1	0
2	0	0	0	0
3	0	1	0	1
4	0	0	0	0

邻接矩阵

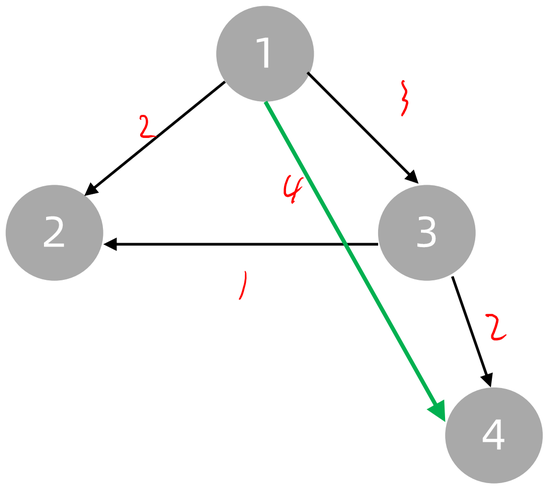
1	2	3
2		
3	2	4
4		

出边数组



邻接表

图的存储

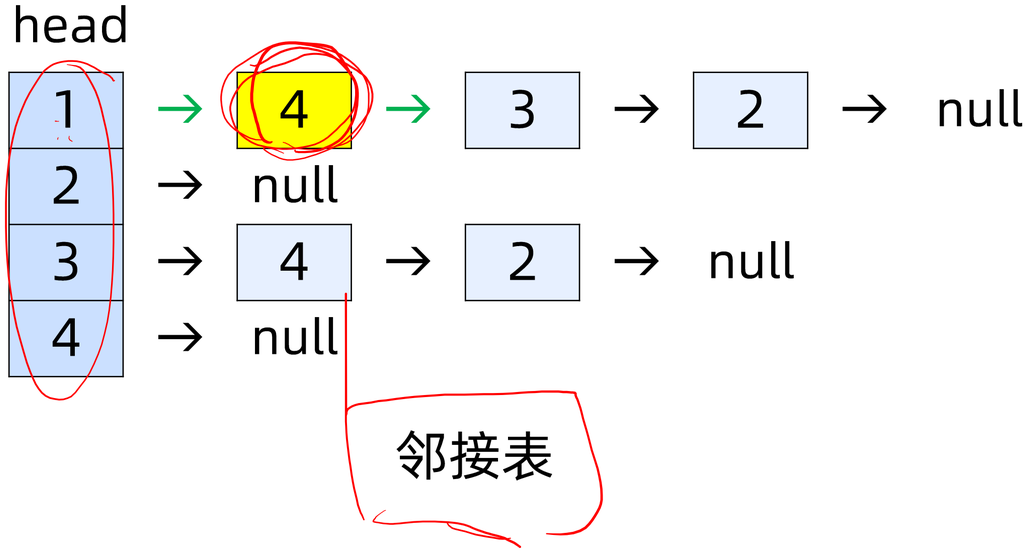


	1	2	3	4
1	0	1	1	1
2	0	0	0	0
3	0	1	0	1
4	0	0	0	0

邻接矩阵

1	2 3 4
2	
3	2 4
4	

出边数组



图的存储

定义

- 邻接矩阵 $O(n^2)$：int graph[MAX_N][MAX_N];
- 出边数组 O(n+m)：vector<int> graph[MAX_N];
- 邻接表 O(n+m)：struct Node { int to; Node* next; };
  Node* head[MAX_N];

新增边 (x,y)

- 邻接矩阵：graph[x][y] = 1;
- 出边数组：graph[x].push_back(y);
- 邻接表：
```
Node* node = new Node();
node->to = y;
node->next = head[x];
head[x] = node;
```

图的遍历

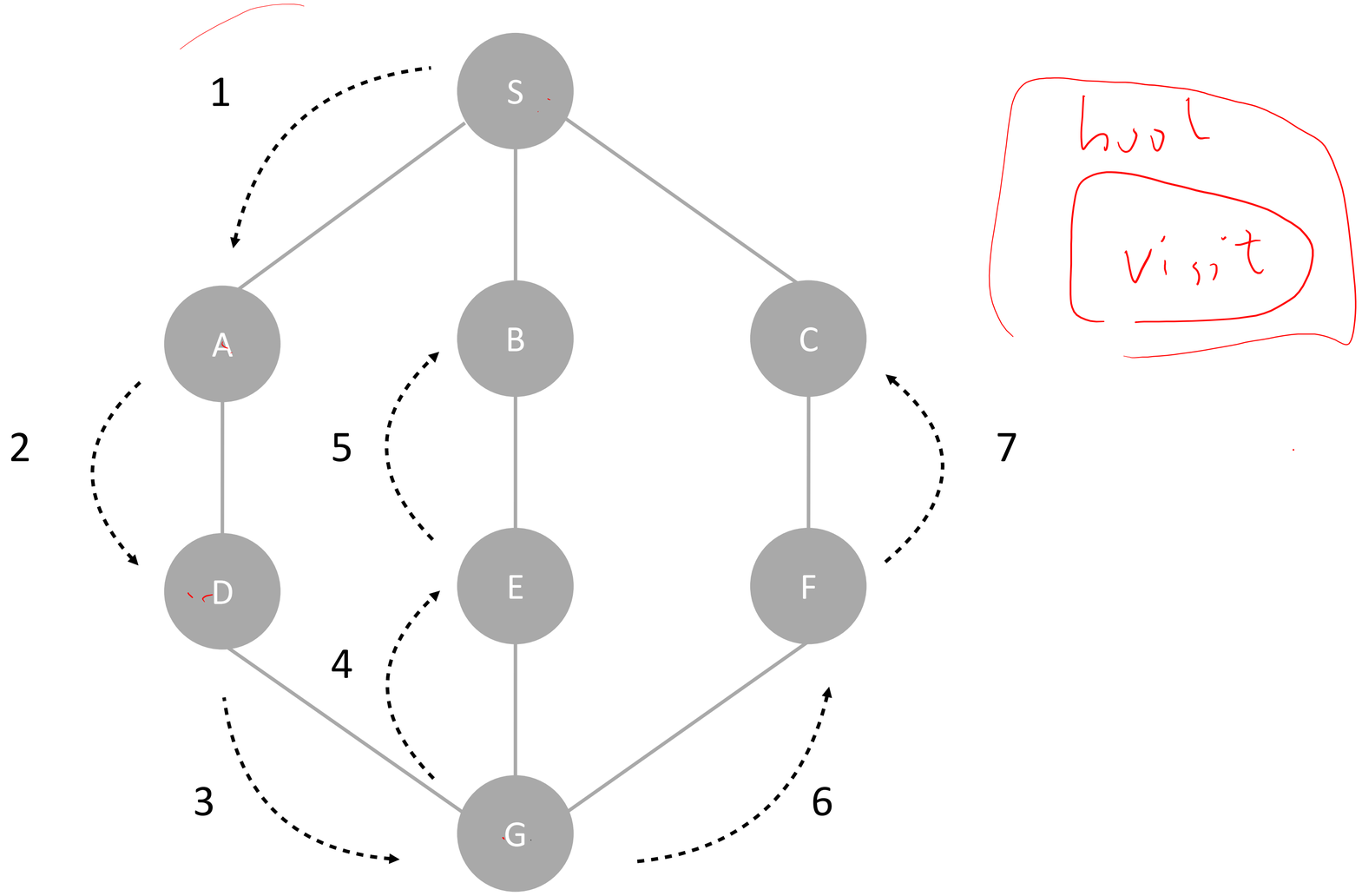
深度优先遍历 无向图 有向图 O(n)

- 划分连通块

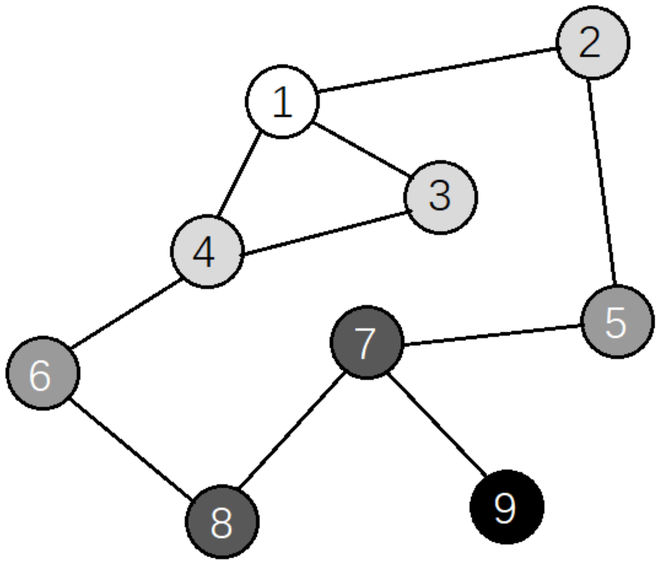
广度优先遍历 → 有向图

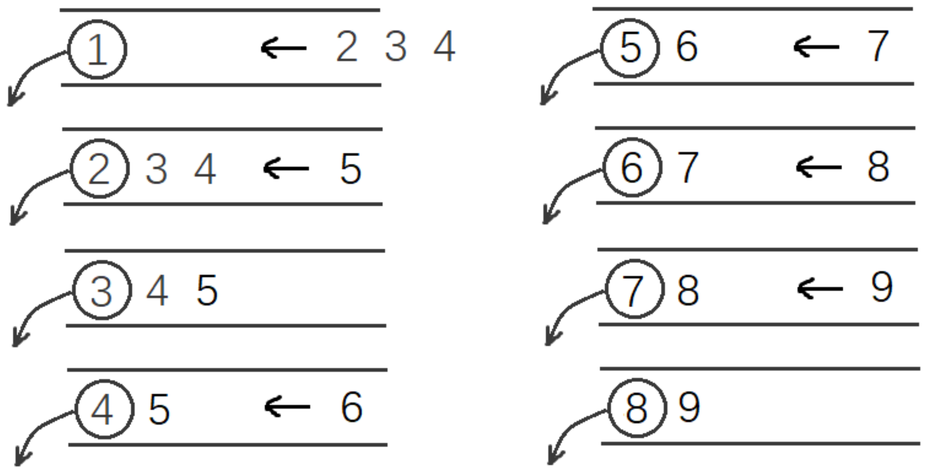
- 拓扑排序

图的深度优先遍历



图的广度优先遍历





实战

课程表

https://leetcode-cn.com/problems/course-schedule/

两种做法：

- 深度优先遍历 - 找环
- 广度优先遍历 - 拓扑排序

课程表 II（Homework）

https://leetcode-cn.com/problems/course-schedule-ii/

实战

冗余连接

https://leetcode-cn.com/problems/redundant-connection/description/

数据实际上是一棵基环树

深度优先遍历找环，环上删除一条边

冗余连接 II（选做）

https://leetcode-cn.com/problems/redundant-connection-ii/

THANKS