玩儿转图论算法

liuyubobobo

遍历的意义

liuyubobobo

遍历的意义

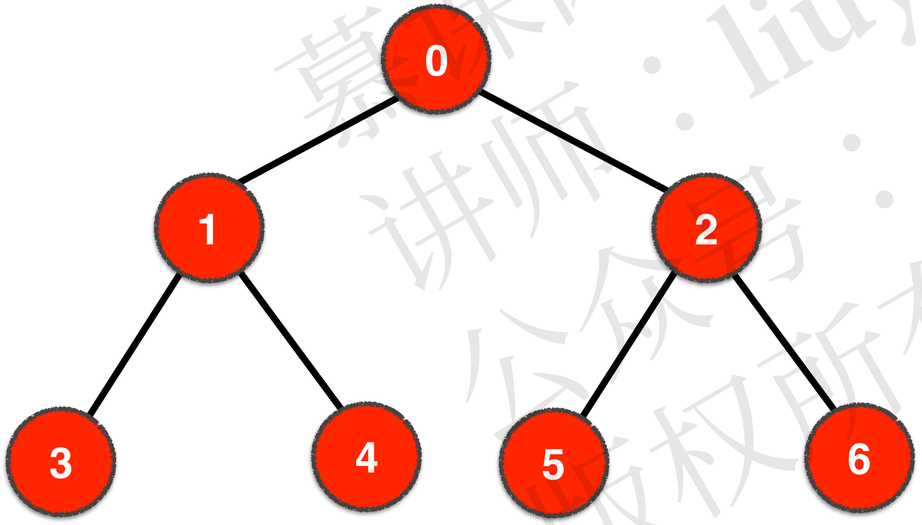
图是一种数据结构

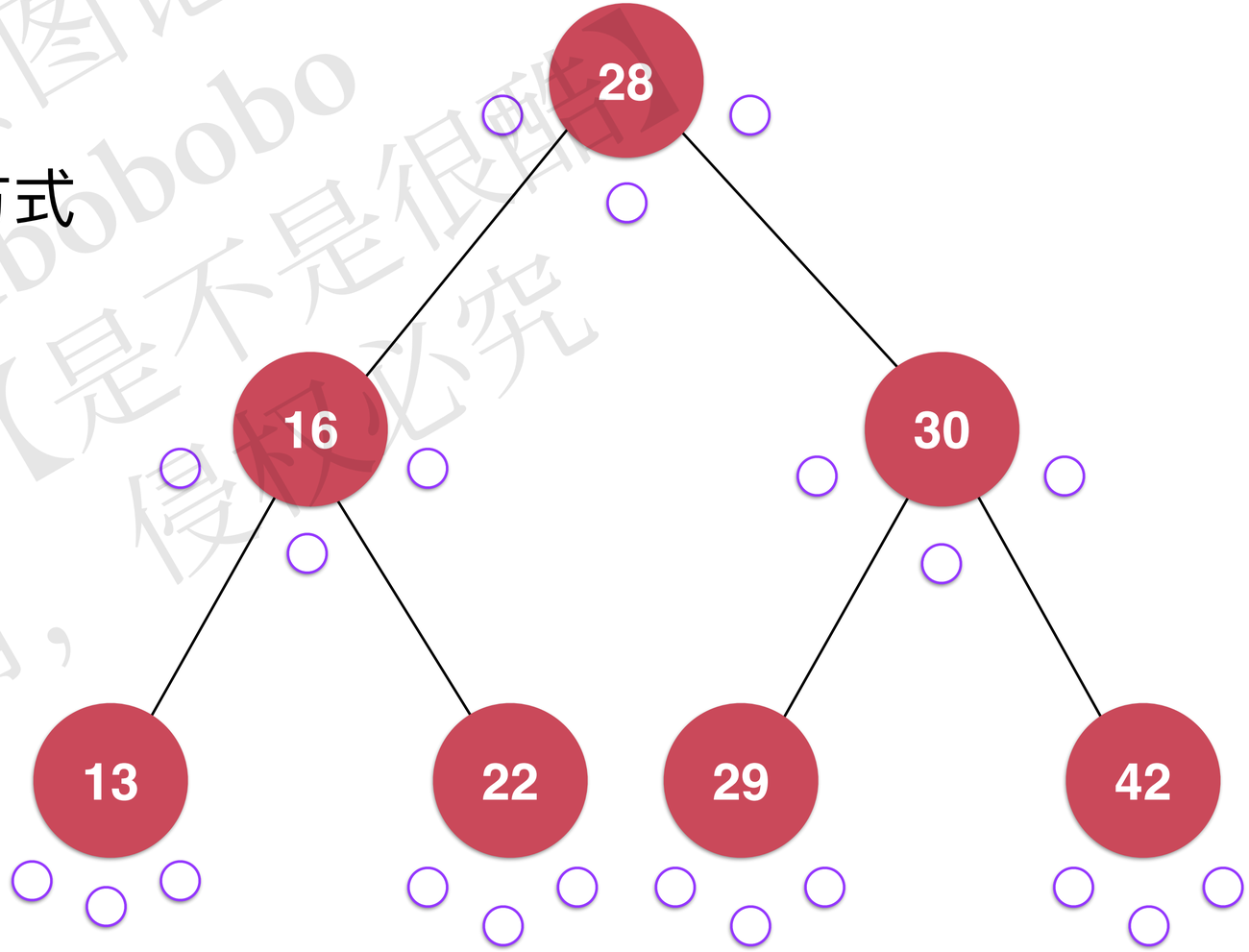
每种数据结构，都必须有“遍历”的方式

遍历的意义

图是一种数据结构

每种数据结构，都必须有“遍历”的方式

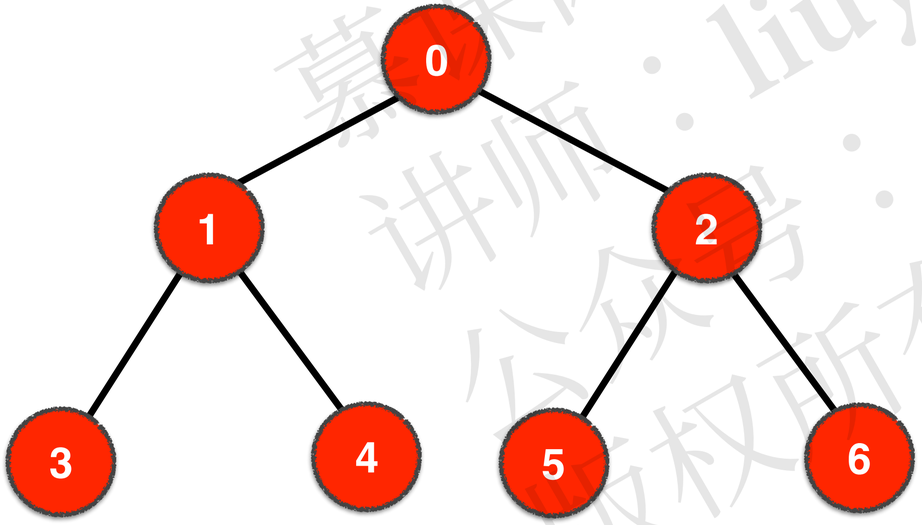




遍历的意义

图是一种数据结构

每种数据结构，都必须有“遍历”的方式



前序遍历：0 1 3 4 2 5 6

中序遍历：3 1 4 0 5 2 6

后序遍历：3 4 1 5 6 2 0

层序遍历：0 1 2 3 4 5 6

遍历的意义

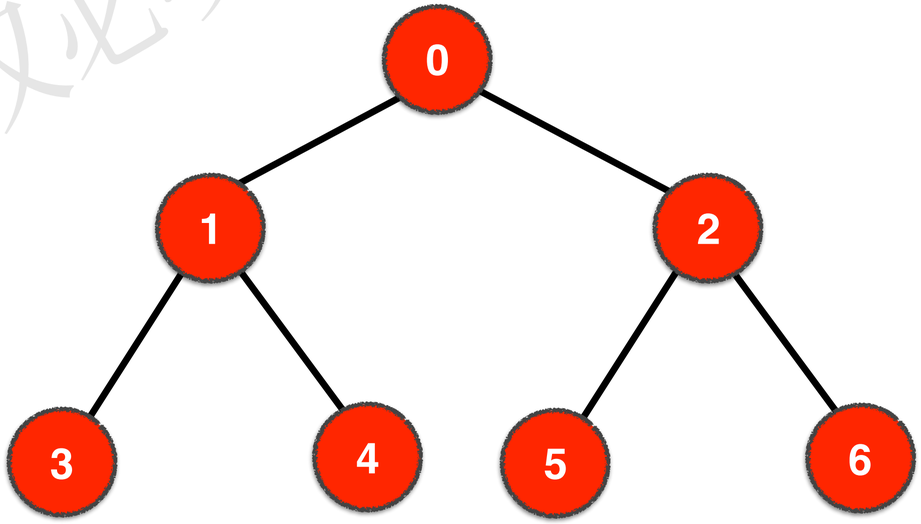
图是一种数据结构

每种数据结构，都必须有“遍历”的方式

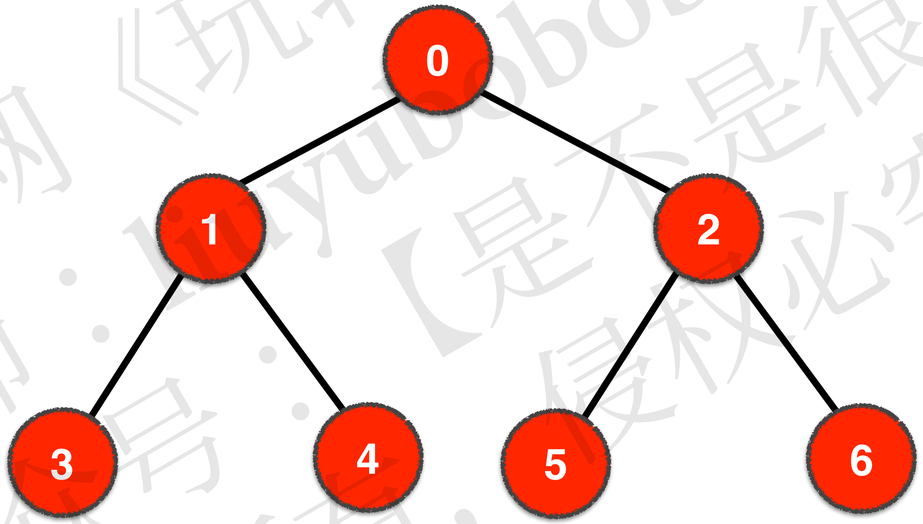
很多算法，本质都是“遍历”

对于图论问题来说，真正理解“遍历”，
已经可以应付80%的面试问题了

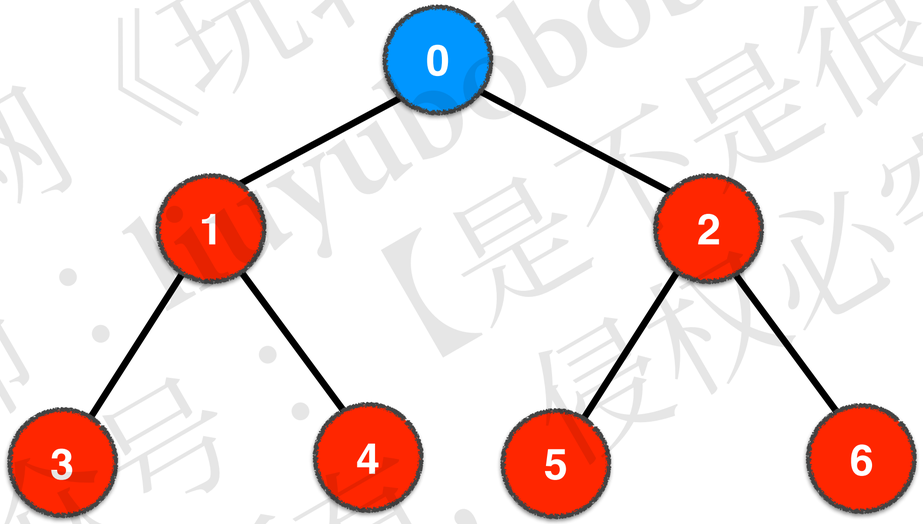
图可以深度遍历，也可以广度遍历



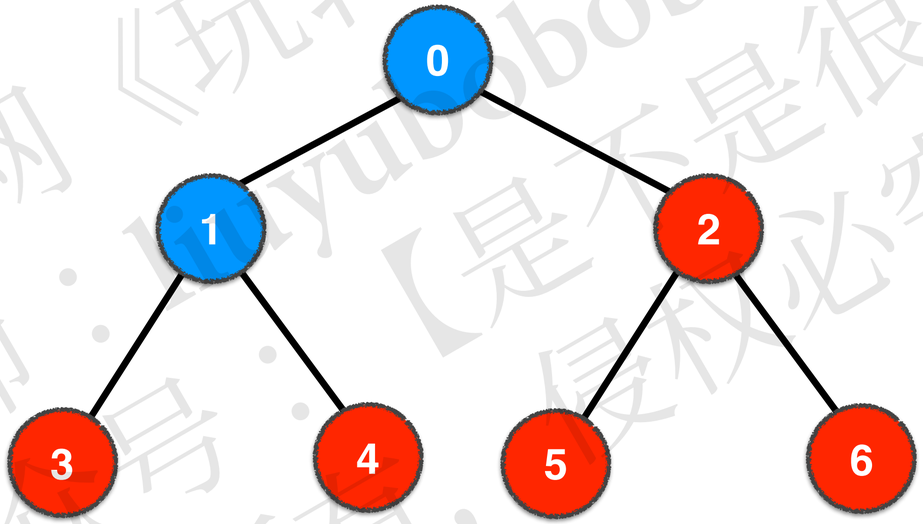
遍历的意义



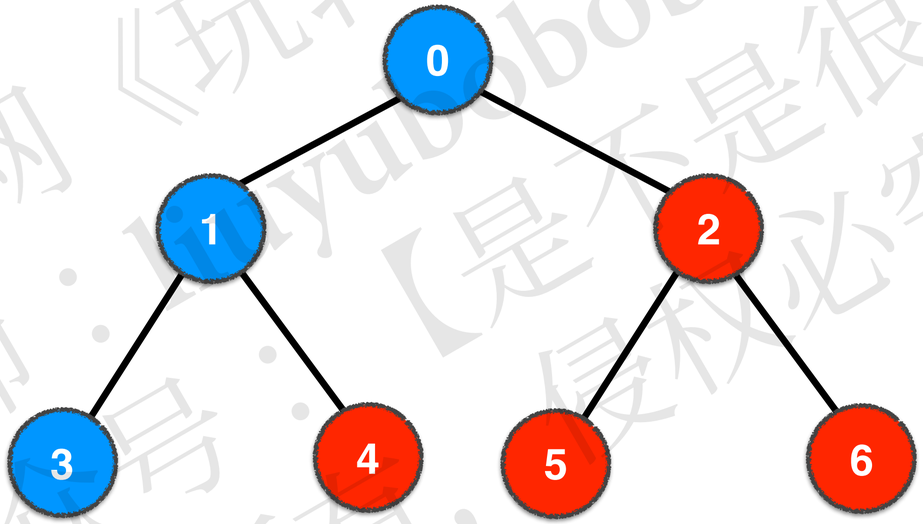
遍历的意义



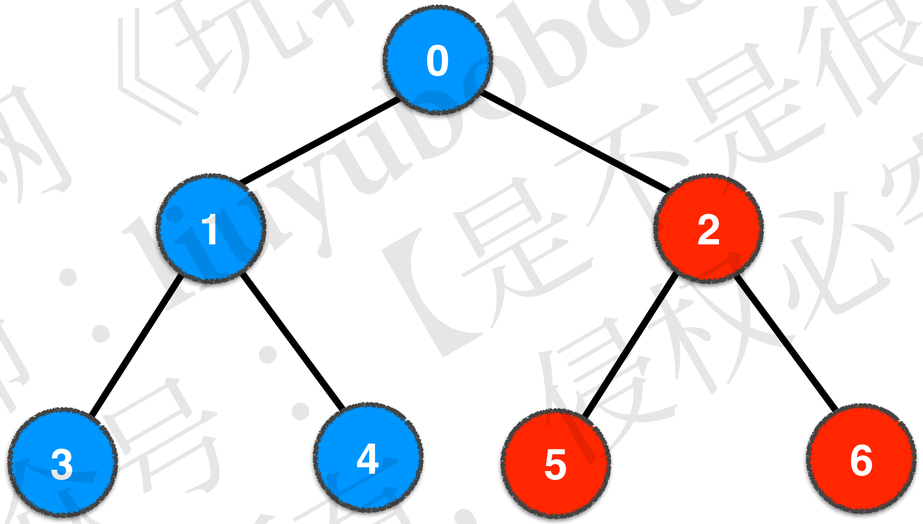
遍历的意义



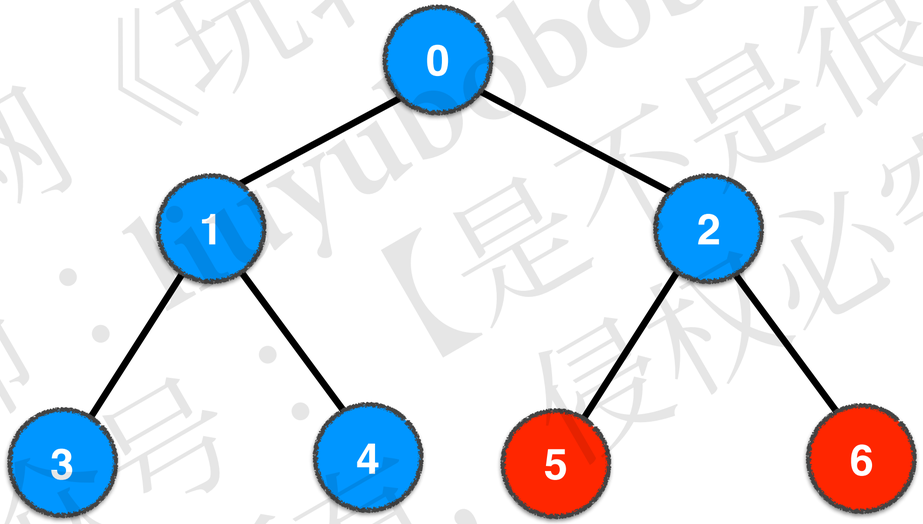
遍历的意义



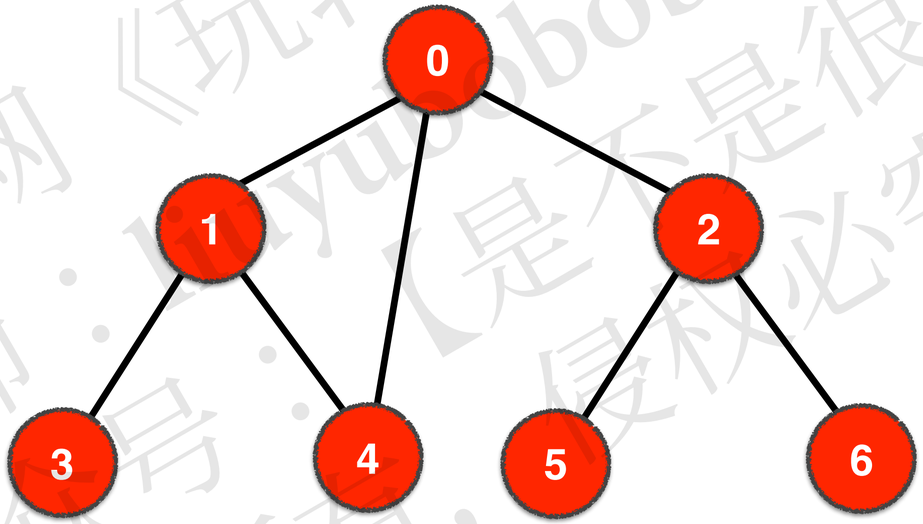
遍历的意义



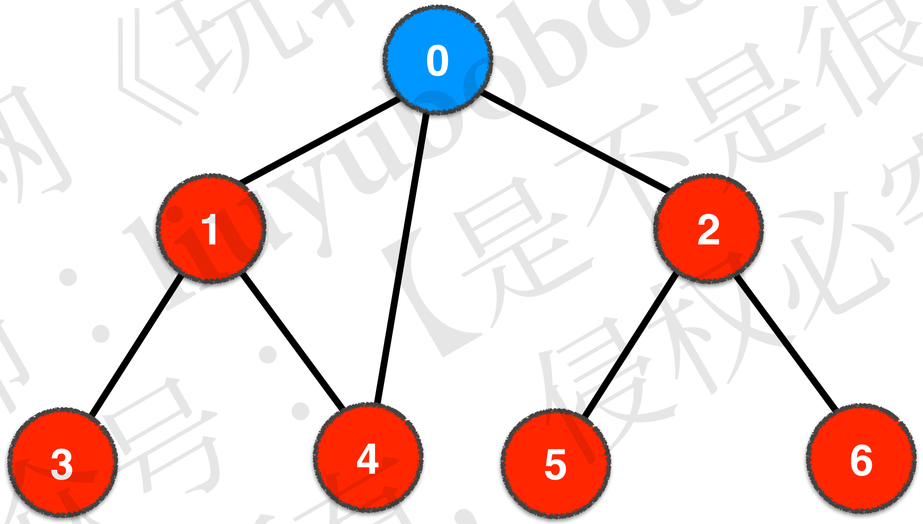
遍历的意义



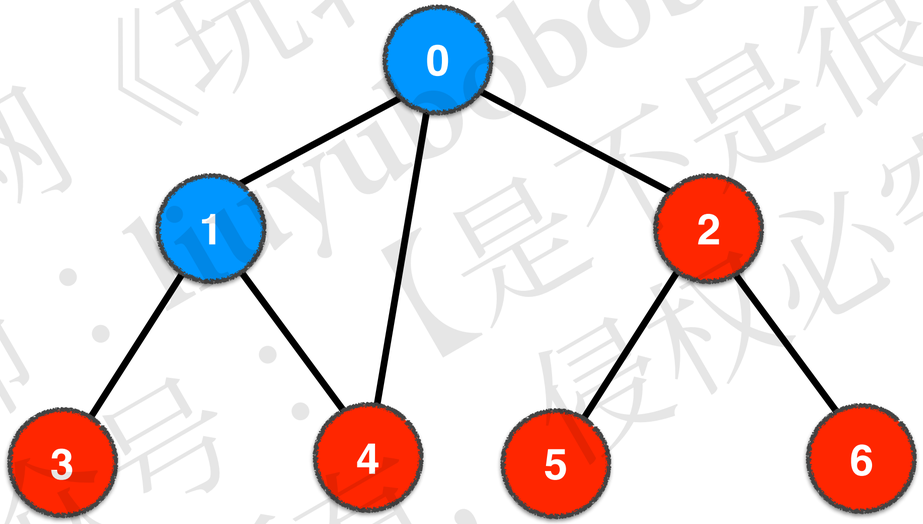
遍历的意义



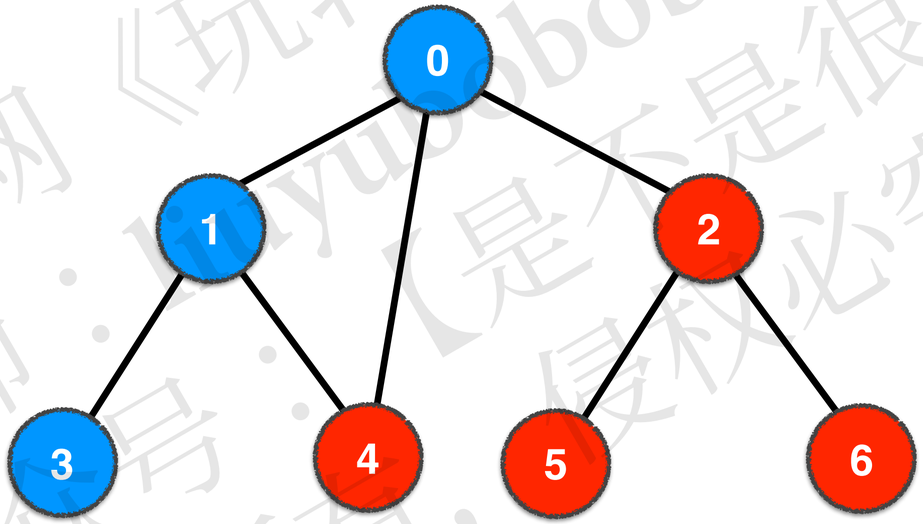
遍历的意义



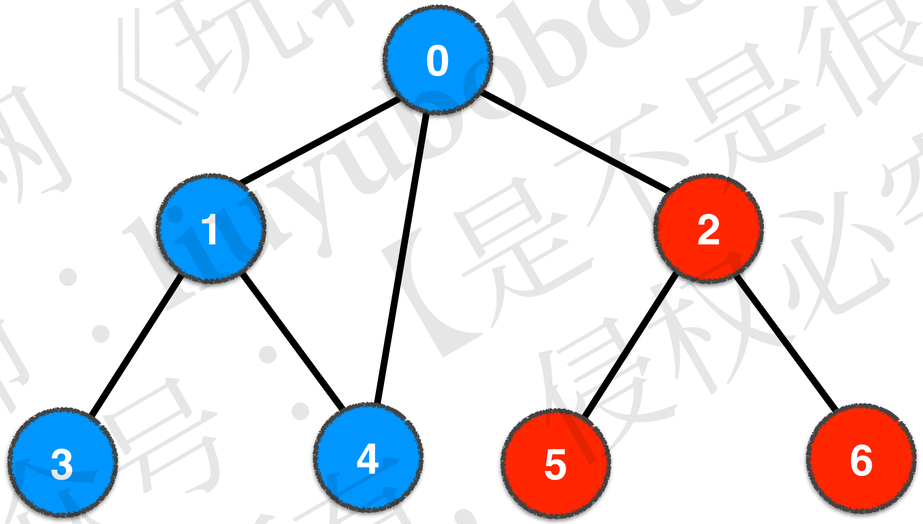
遍历的意义



遍历的意义



遍历的意义



需要记录，哪个节点被遍历了

遍历的意义

强烈建议大家回顾：

Leetcode 144. Binary Tree Preorder Traversal

Leetcode 94. Binary Tree Inorder Traversal　　不强调非递归

Leetcode 145. Binary Tree Postorder Traversal

Leetcode 102. Binary Tree Level Order Traversal

遍历的意义

强烈建议大家回顾：

Leetcode 589. N-ary Tree Preorder Traversal

Leetcode 590. N-ary Tree Postorder Traversal

为什么n叉树的遍历没有中序?

Leetcode 429. N-ary Tree Level Order Traversal

图的深度优先遍历

liuyubobobo

图的深度优先遍历

```
preorderTraversal(root);


preorder(TreeNode node){
    if(node != null){
        list.add(node.val);
        preorder(node.left);
        preorder(node.right);
    }
}
```

遍历

访问所有子树

图的深度优先遍历

```
preorderTraversal(root);


preorder(TreeNode node){
    if(node != null){
        list.add(node.val);
        preorder(node.left);
        preorder(node.right);
    }
}
```

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
    visited[v] = true;
    list.add(v);
    for(int w: adj(v))
        if(!visited[w])
            dfs(w);
}
```

DFS逻辑的微观解读

liuyubobobo

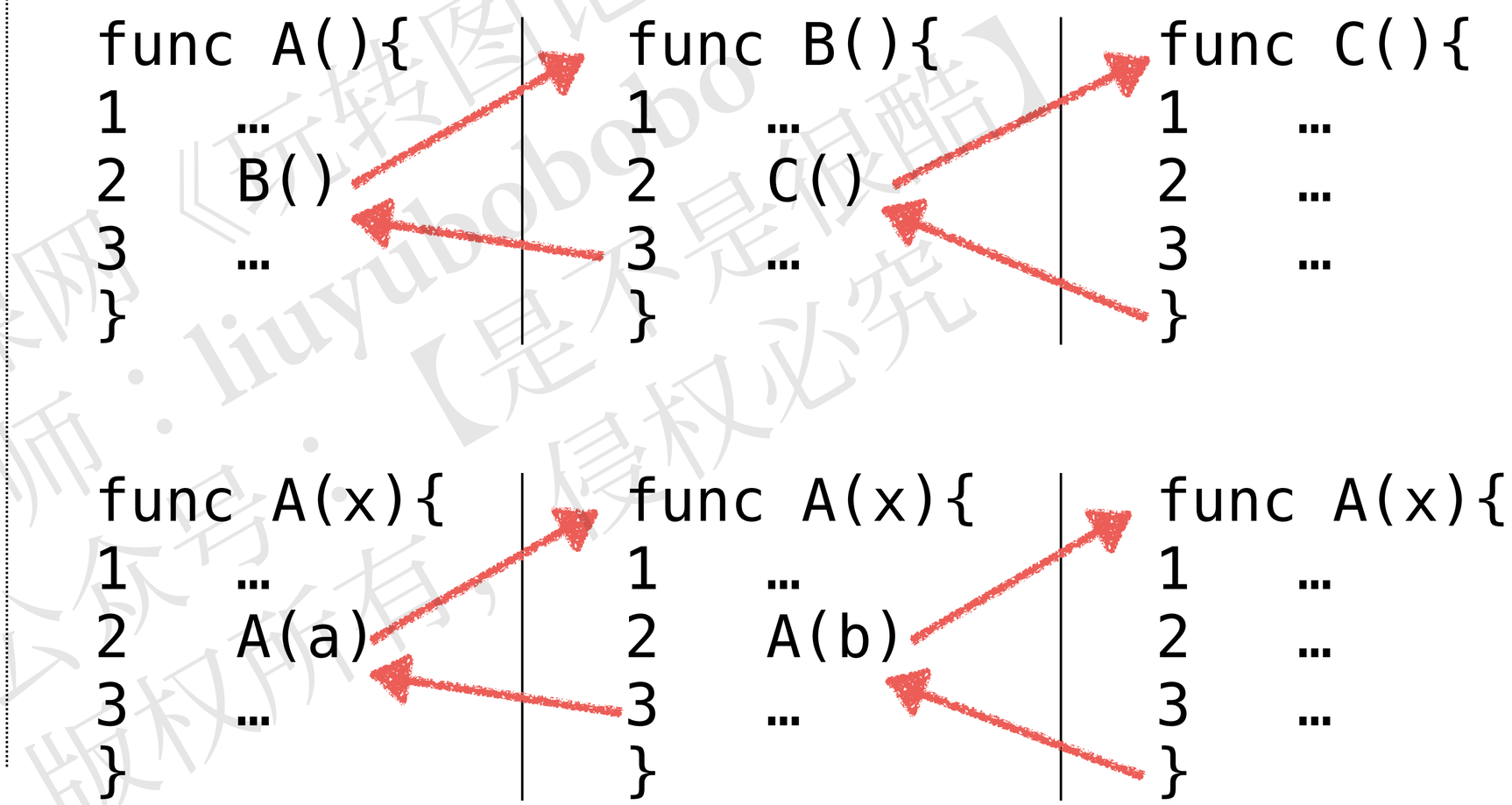
图的深度优先遍历

```
preorderTraversal(root);


preorder(TreeNode node){
    if(node != null){
        list.add(node.val);
        preorder(node.left);
        preorder(node.right);
    }
}
```

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
    visited[v] = true;
    list.add(v);
    for(int w: adj(v))
        if(!visited[w])
            dfs(w);
}
```

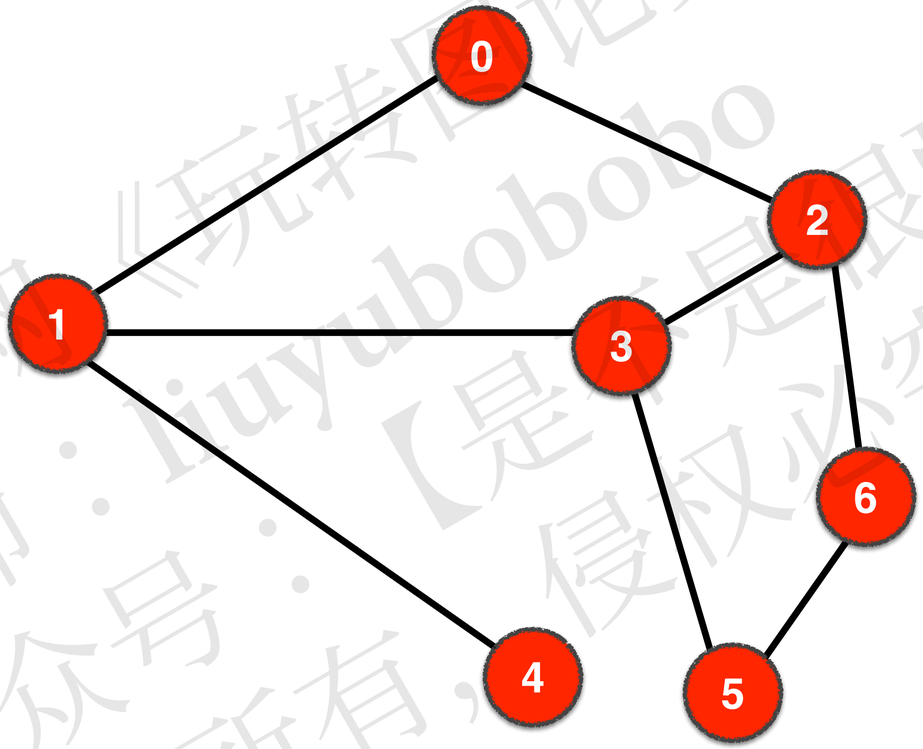
图的深度优先遍历

```
visited[0…V-1] = false;

dfs(0);

dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```



图的深度优先遍历

```
visited[0…V-1] = false;

dfs(0);

dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```



visited	0	1	2	3	4	5	6

遍历结果:

0: 1 2

1: 0 3 4

2: 0 3 6

3: 1 2 5

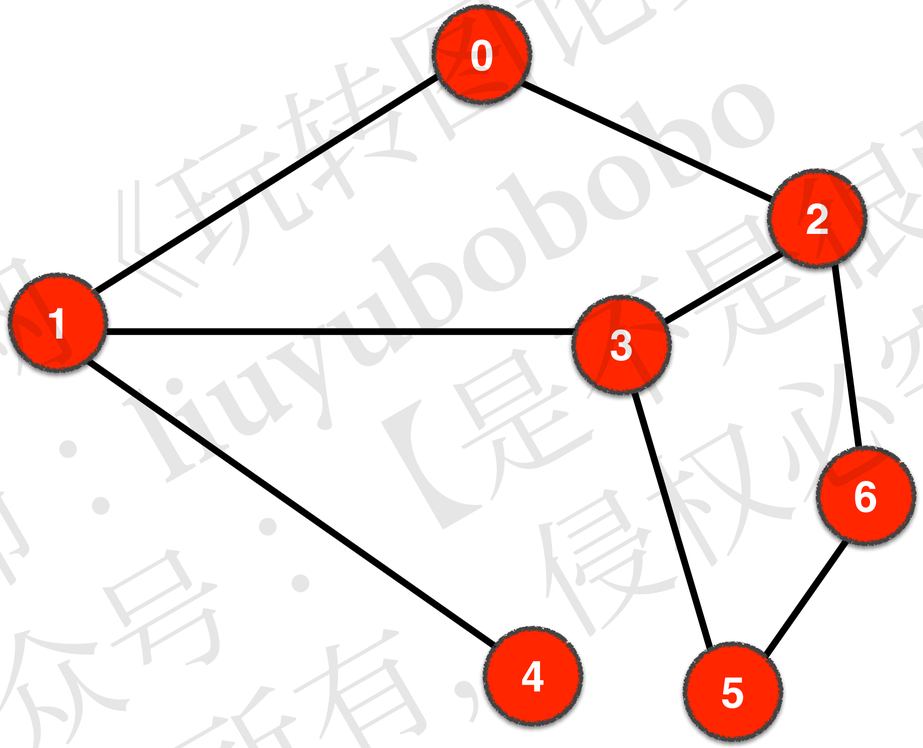
4: 1

5: 3 6

6: 2 5

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
dfs(0)
```



visited	0	1	2	3	4	5	6

遍历结果:

0: 1 2

1: 0 3 4

2: 0 3 6

3: 1 2 5

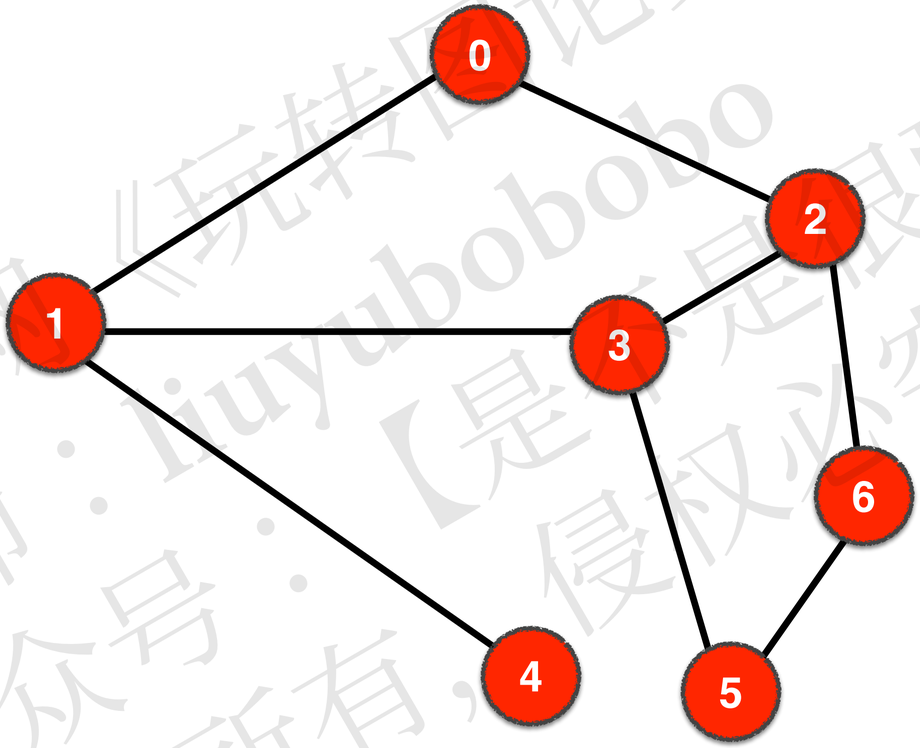
4: 1

5: 3 6

6: 2 5

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
dfs(0)
```



visited	0	1	2	3	4	5	6

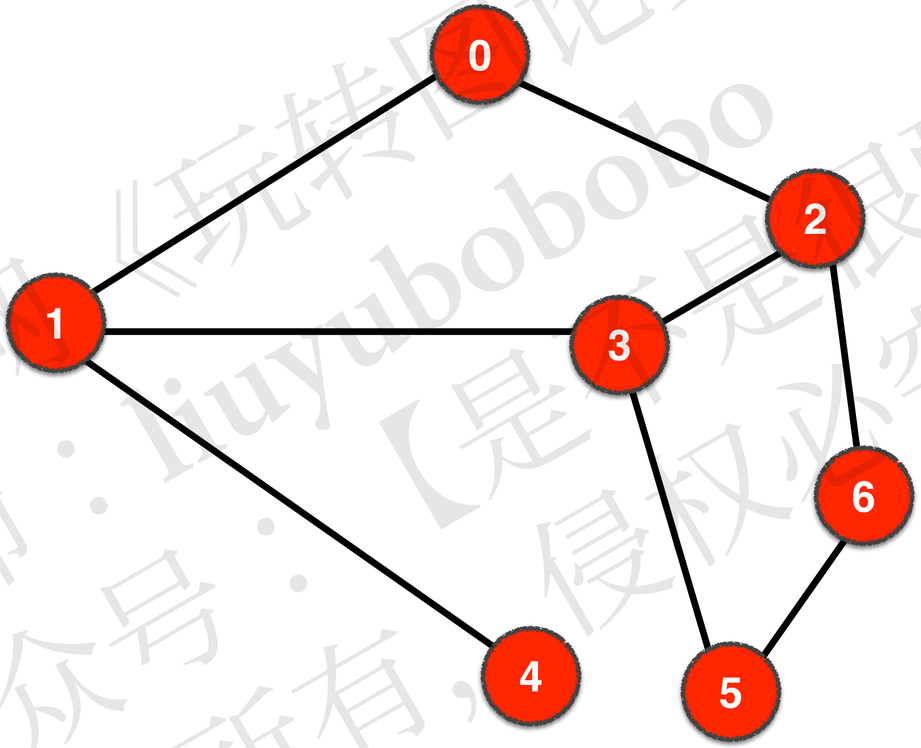
遍历结果：**0**

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1)



visited	0	1	2	3	4	5	6

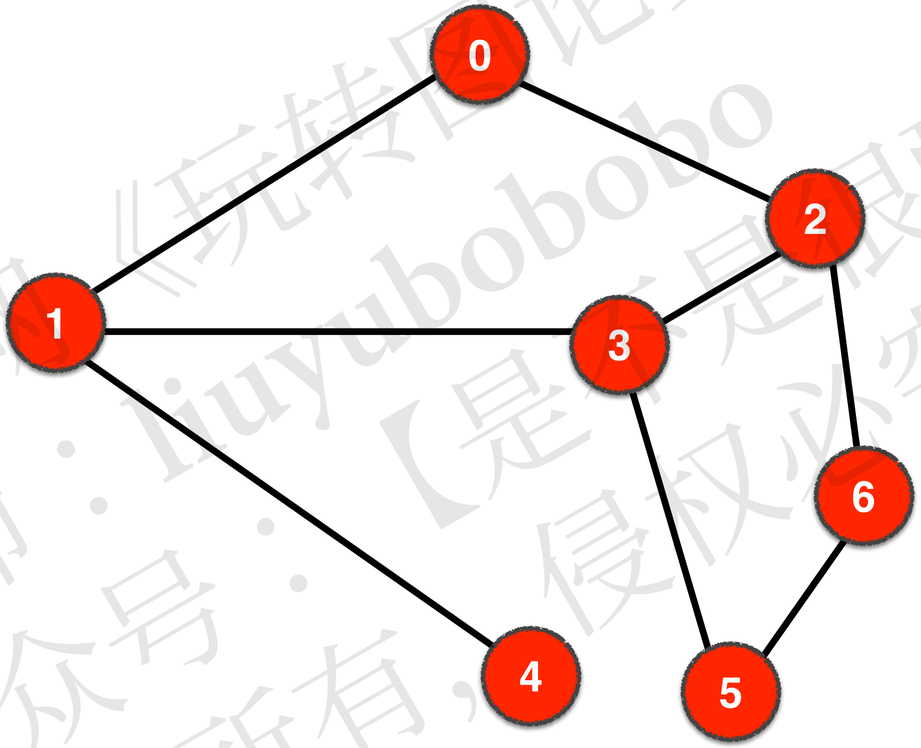
遍历结果：**0**

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1)



visited

0	1	2	3	4	5	6

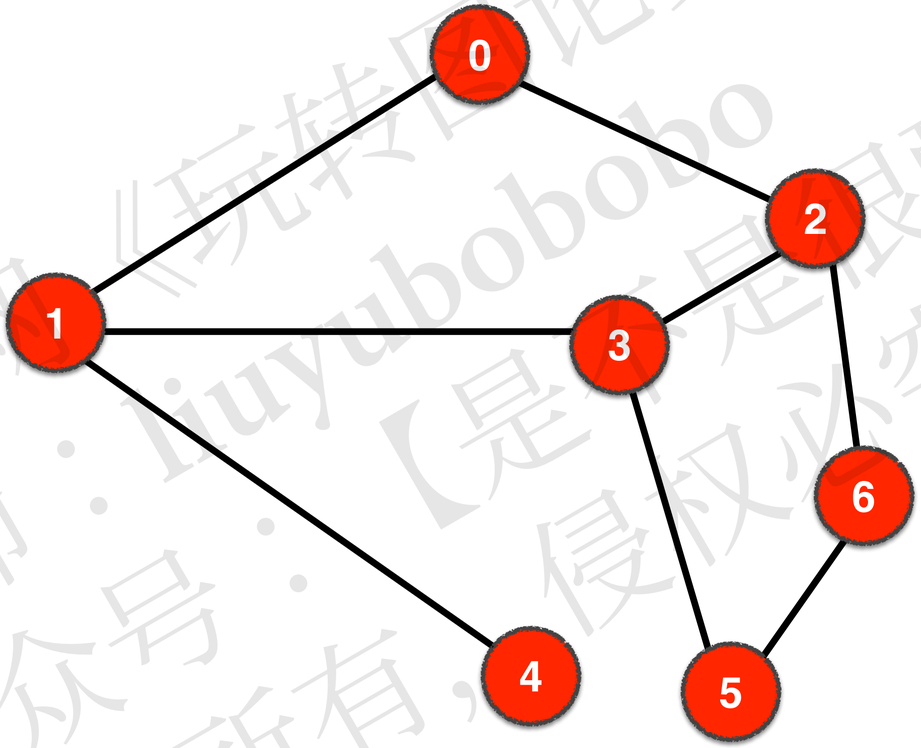
遍历结果：**0 1**

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1)



visited	0	1	2	3	4	5	6

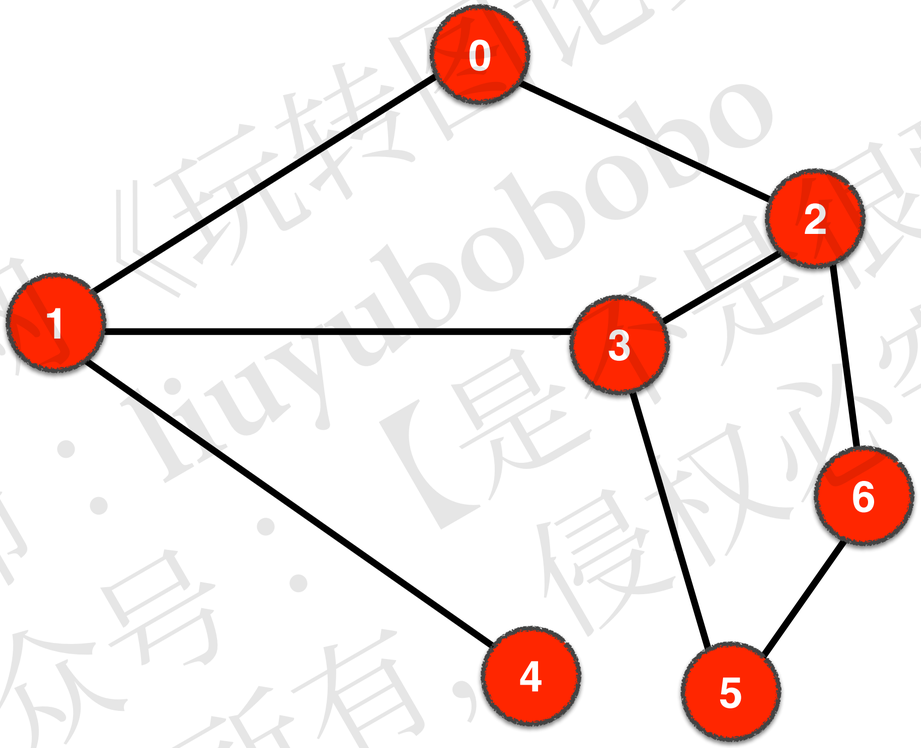
遍历结果：**0 1**

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3)



visited	0	1	2	3	4	5	6

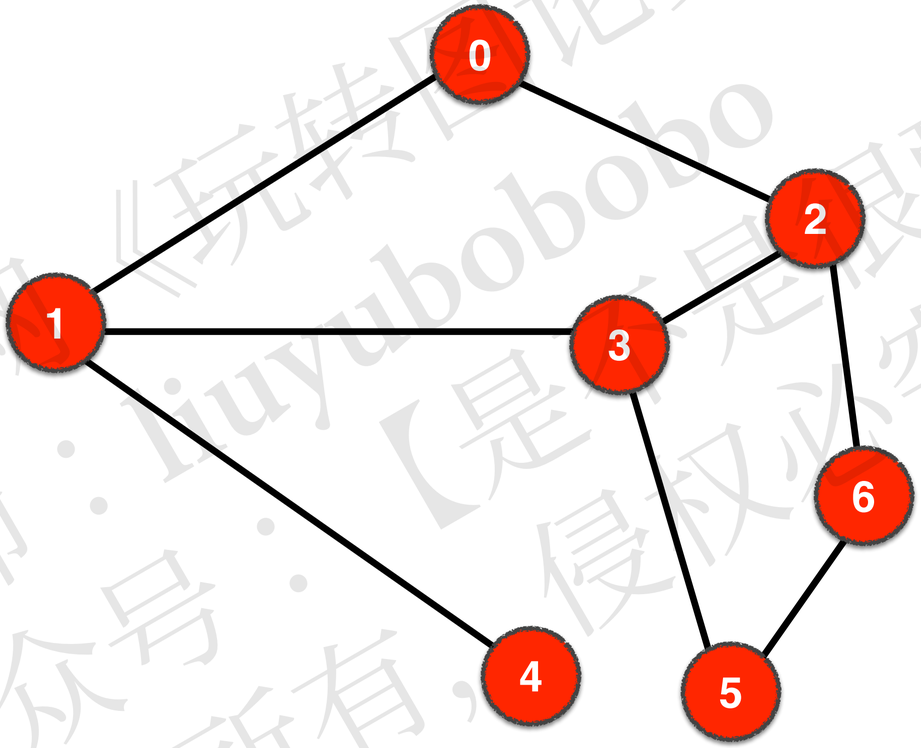
遍历结果：**0 1**

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3)



visited	0	1	2	3	4	5	6

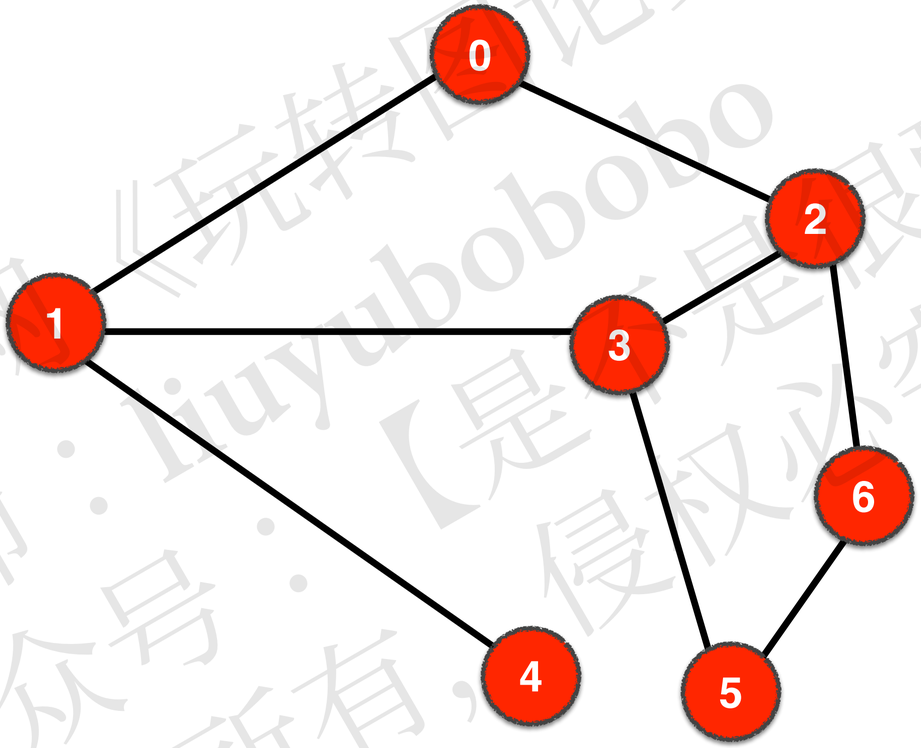
遍历结果：**0 1 3**

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3)



visited	0	1	2	3	4	5	6

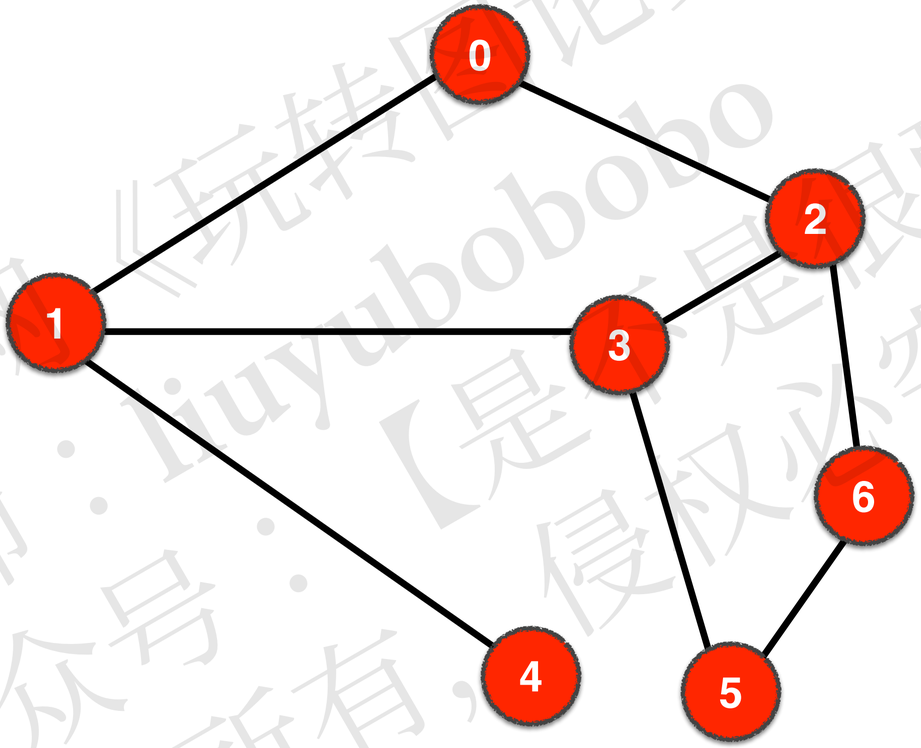
遍历结果：0 1 3

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2)



visited	0	1	2	3	4	5	6

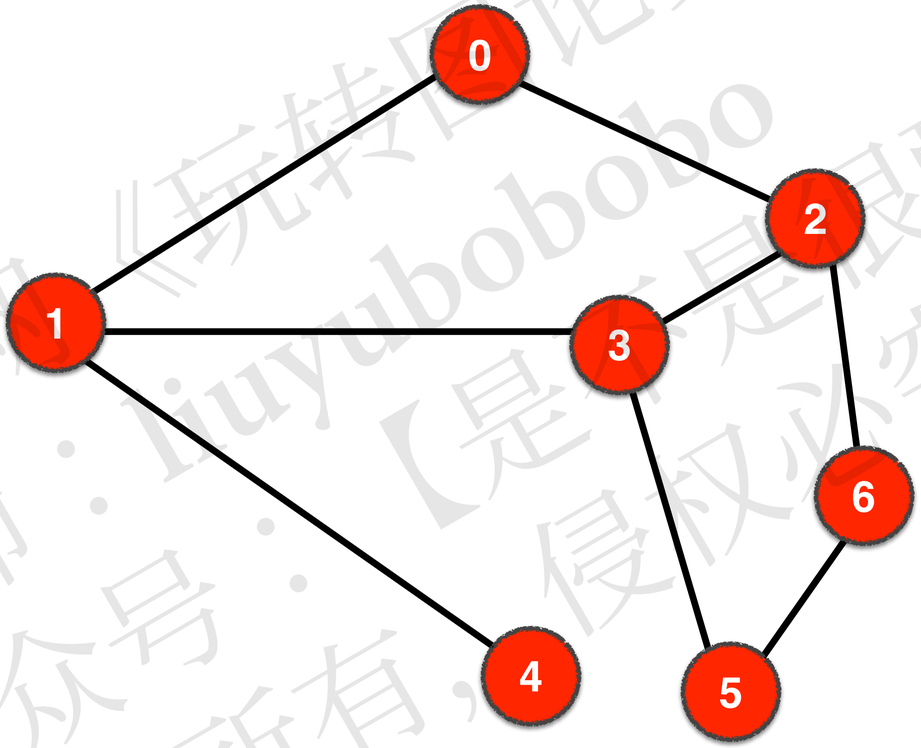
遍历结果：**0 1 3**

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2)



visited	0	1	2	3	4	5	6

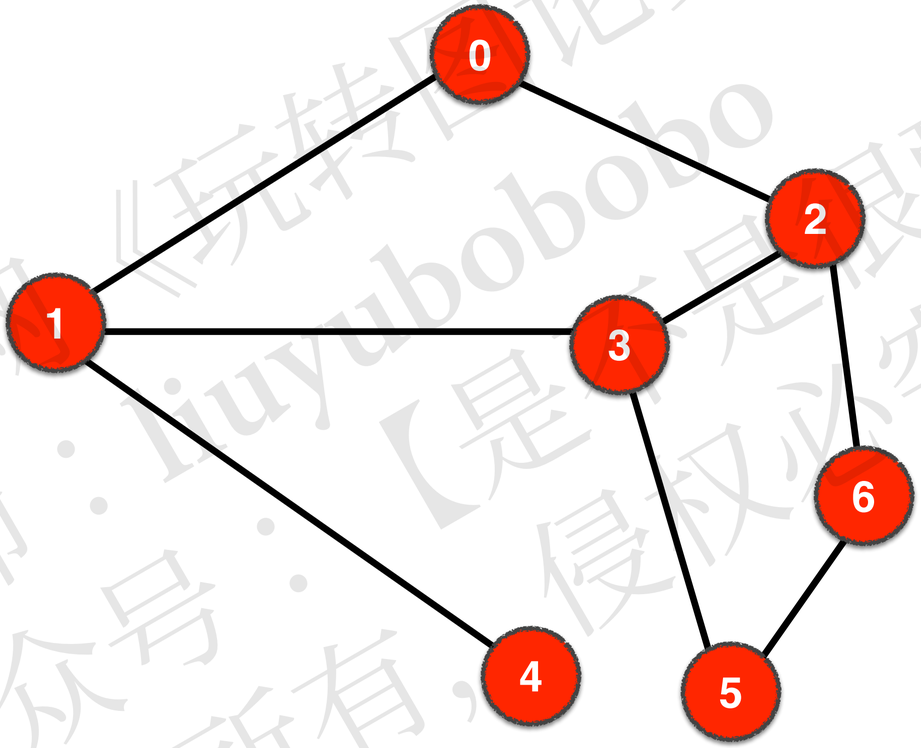
遍历结果：0 1 3 2

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2)



visited	0	1	2	3	4	5	6

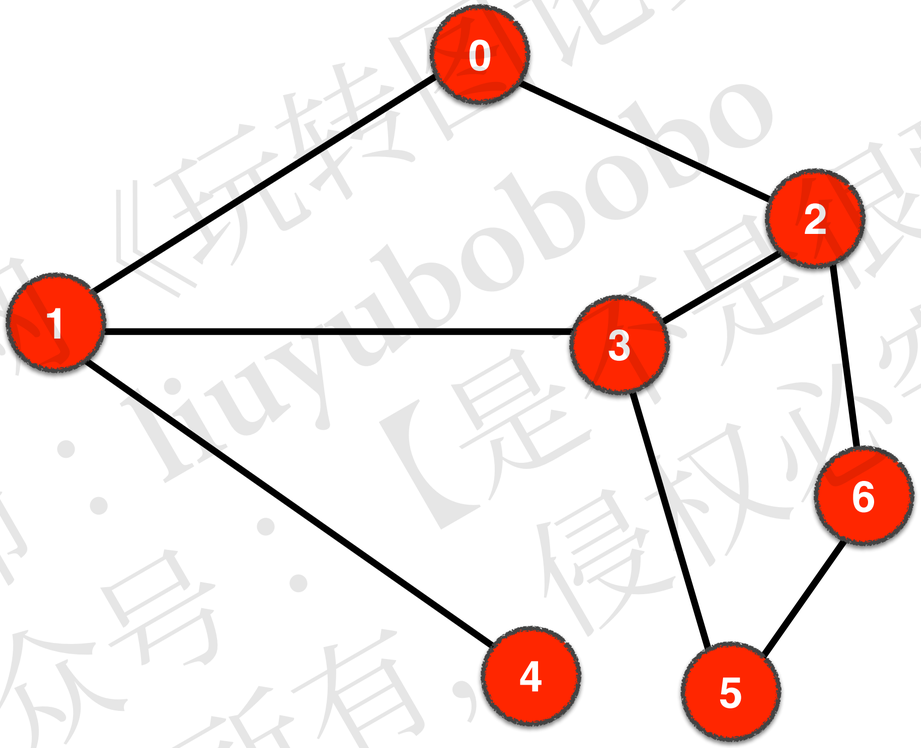
遍历结果：0 1 3 2

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2)



visited	0	1	2	3	4	5	6

遍历结果：0 1 3 2

0: 1 2

1: 0 3 4

2: 0 3 6

3: 1 2 5

4: 1

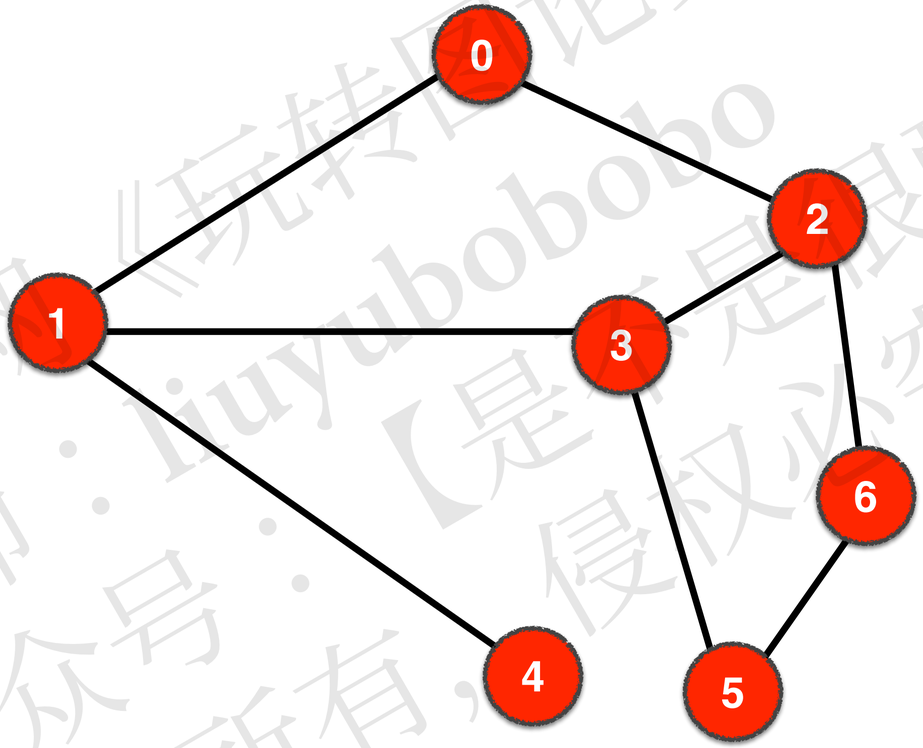
5: 3 6

6: 2 5

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2) ▶ dfs(6)



visited	0	1	2	3	4	5	6

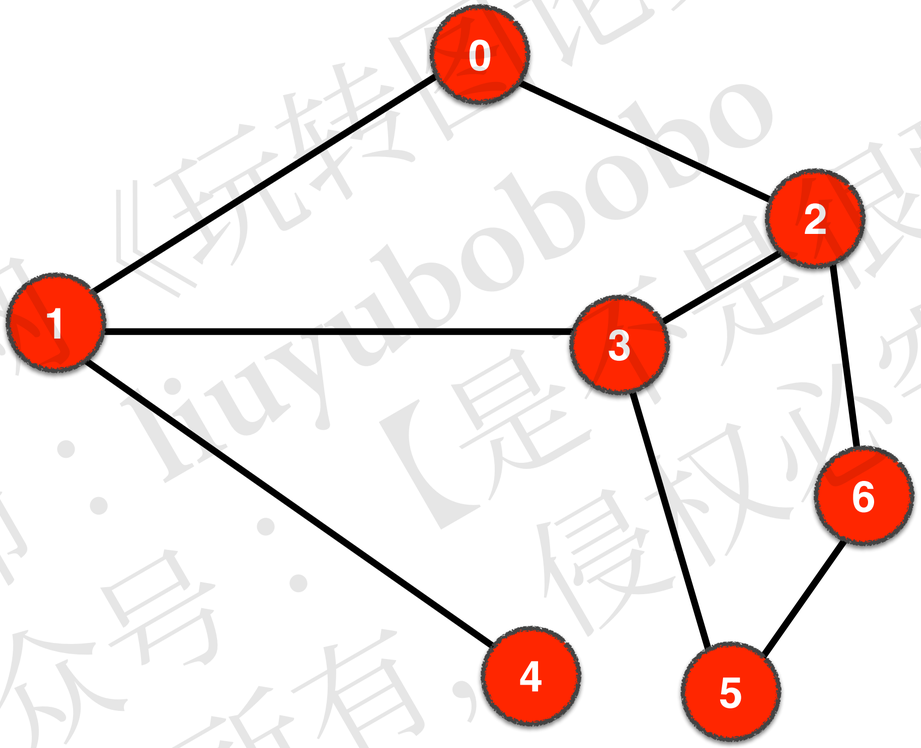
遍历结果：0 1 3 2

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2) ▶ dfs(6)



visited	0	1	2	3	4	5	6

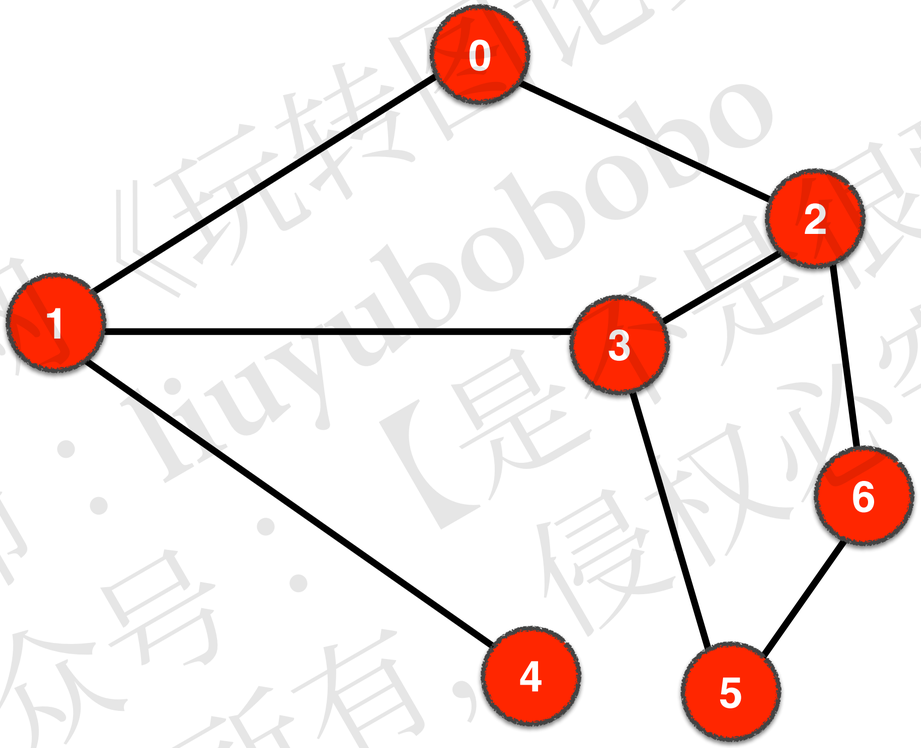
遍历结果：0 1 3 2 6

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2) ▶ dfs(6)



visited	0	1	2	3	4	5	6

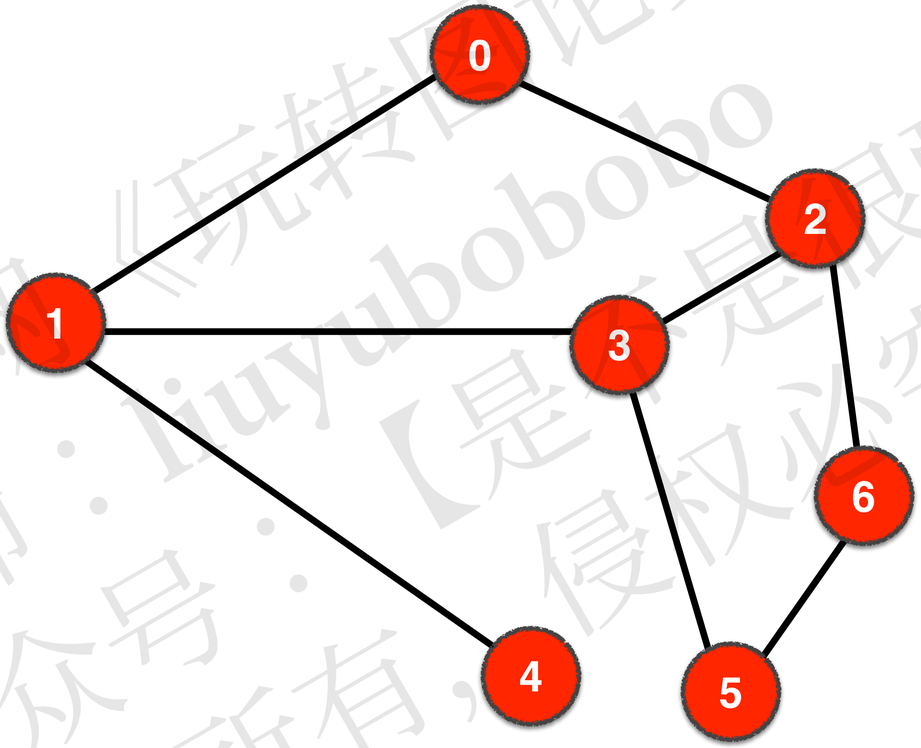
遍历结果：0 1 3 2 6

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2) ▶ dfs(6) ▶ dfs(5)



visited	0	1	2	3	4	5	6

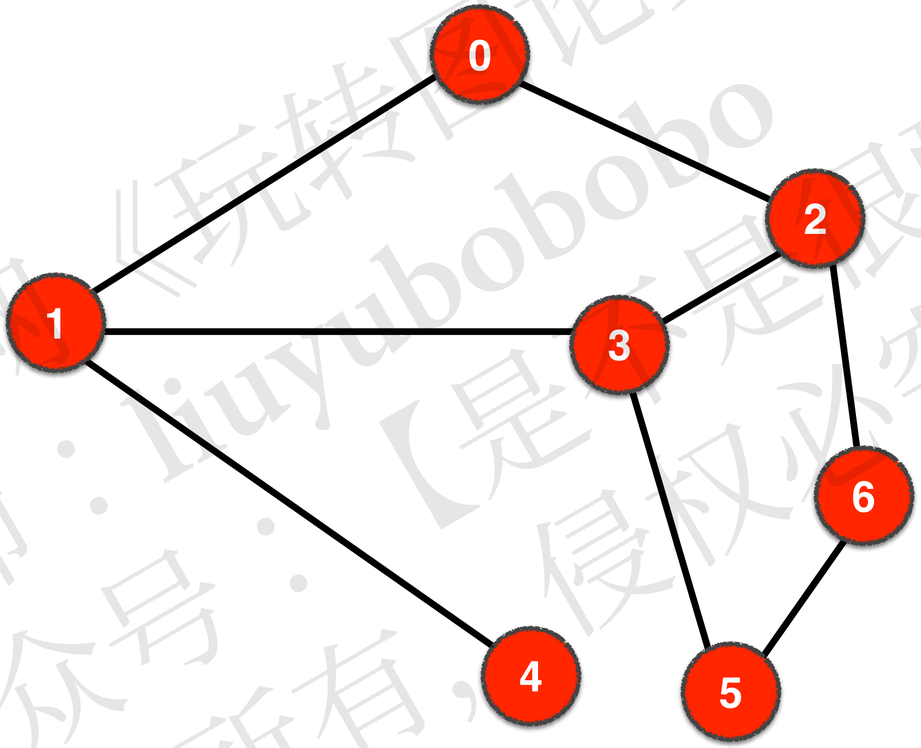
遍历结果：0 1 3 2 6

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2) ▶ dfs(6) ▶ dfs(5)



visited	0	1	2	3	4	5	6

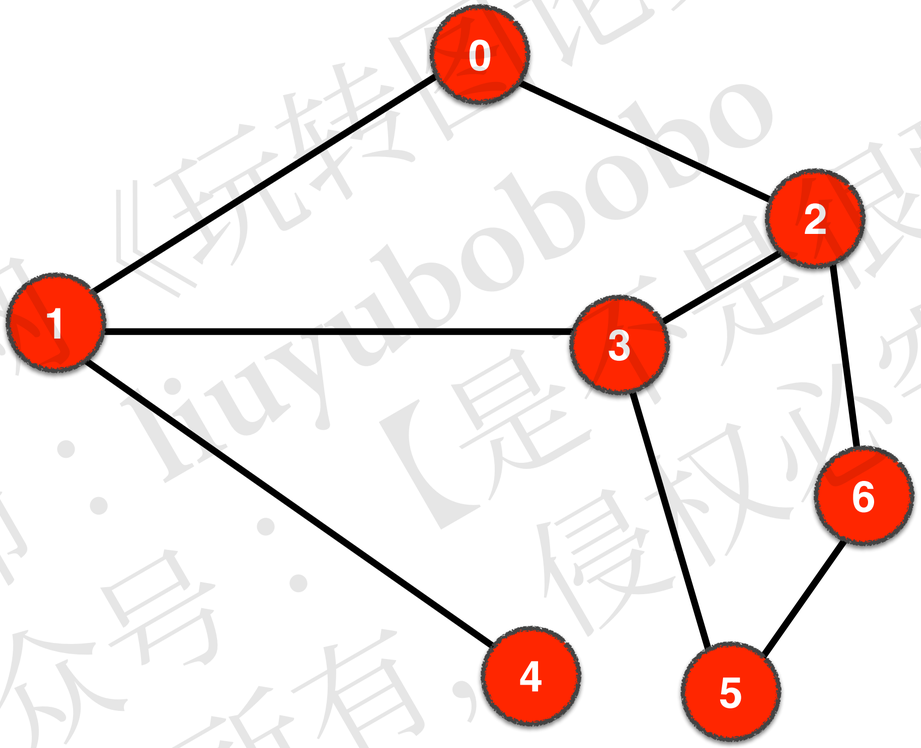
遍历结果：0 1 3 2 6 5

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2) ▶ dfs(6) ▶ dfs(5)



visited

0	1	2	3	4	5	6

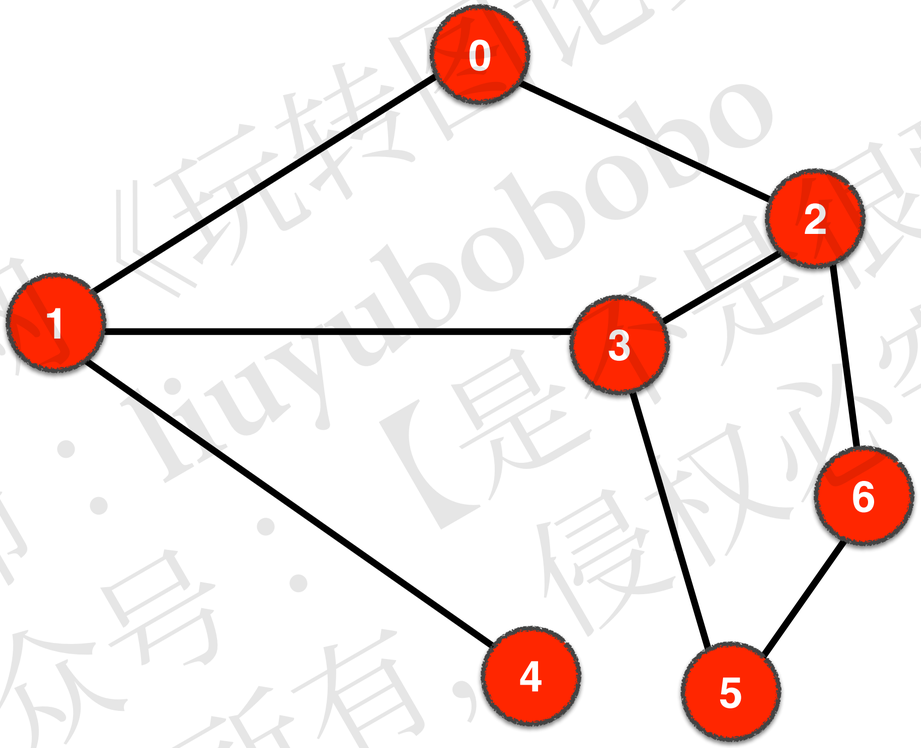
遍历结果：0 1 3 2 6 5

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3) ▶ dfs(2) ▶ dfs(6) ▶ dfs(5)



visited	0	1	2	3	4	5	6

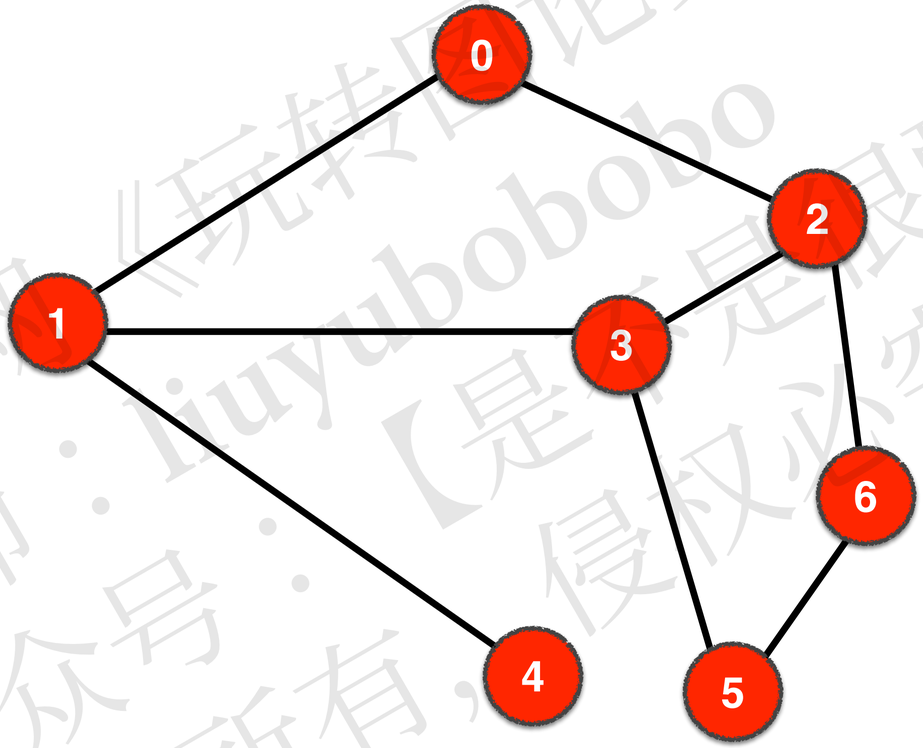
遍历结果：0 1 3 2 6 5

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(3)



visited	0	1	2	3	4	5	6

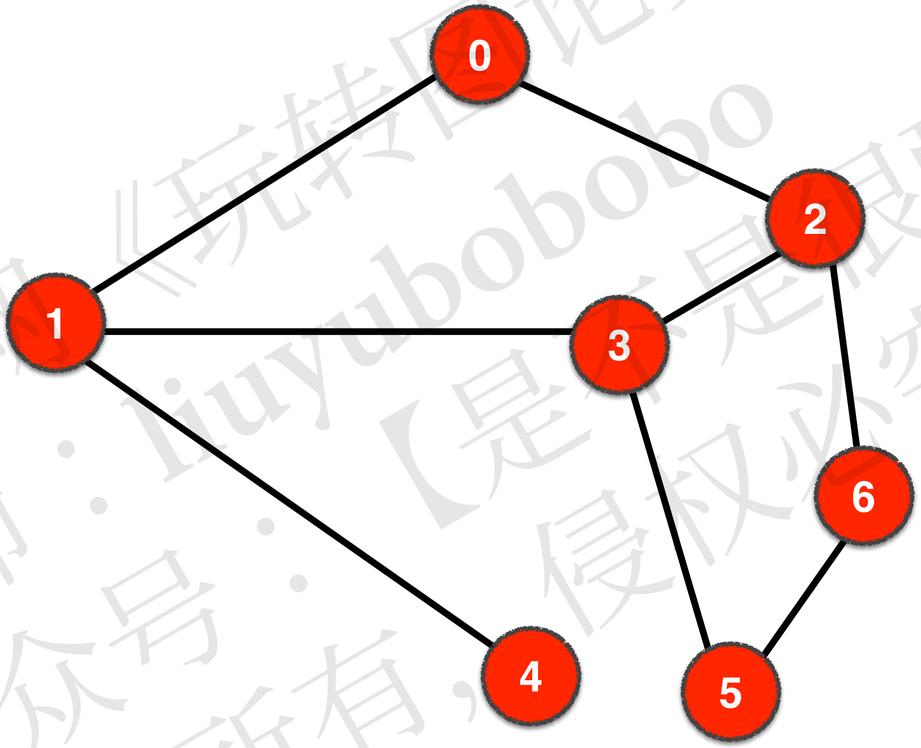
遍历结果：0 1 3 2 6 5

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(4)



visited	0	1	2	3	4	5	6

遍历结果：0 1 3 2 6 5

0: 1 2

1: 0 3 4

2: 0 3 6

3: 1 2 5

4: 1

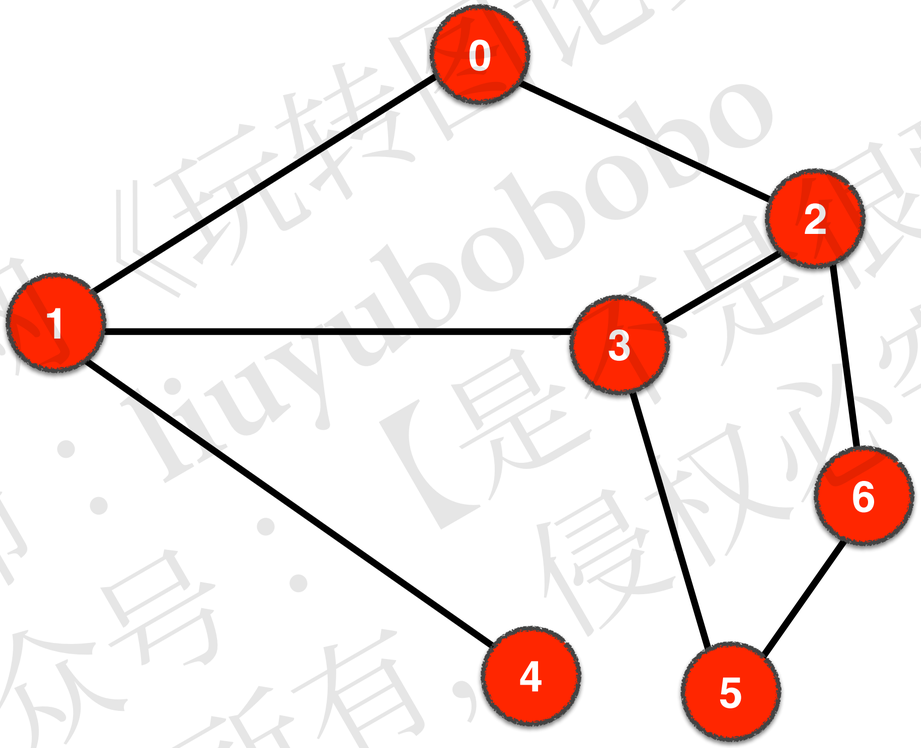
5: 3 6

6: 2 5

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(4)



visited

0	1	2	3	4	5	6

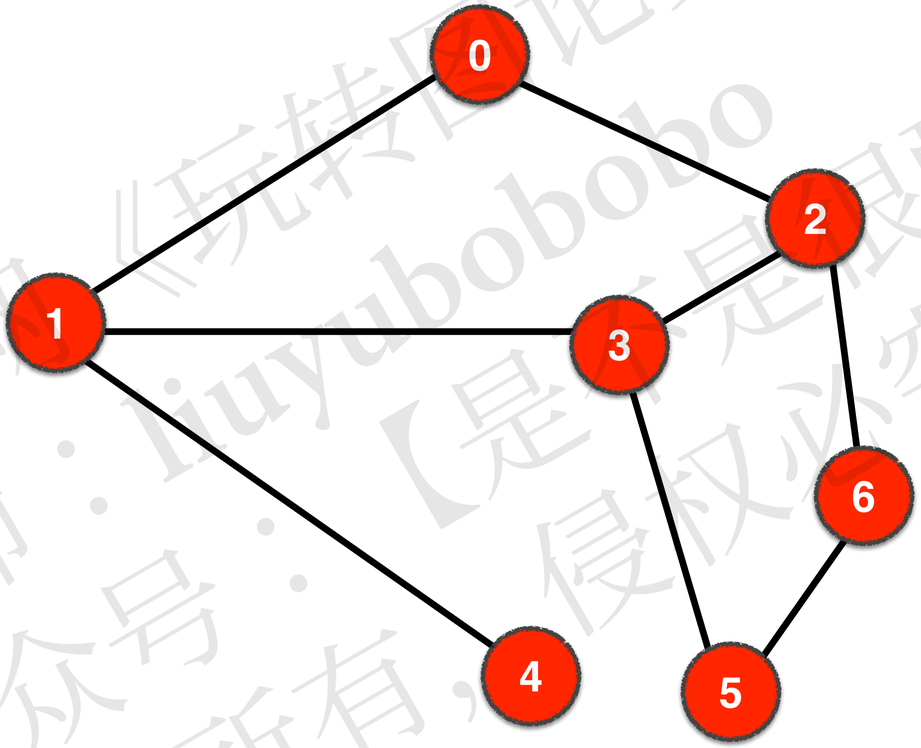
遍历结果：0 1 3 2 6 5 4

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
```

dfs(0) ▶ dfs(1) ▶ dfs(4)



visited

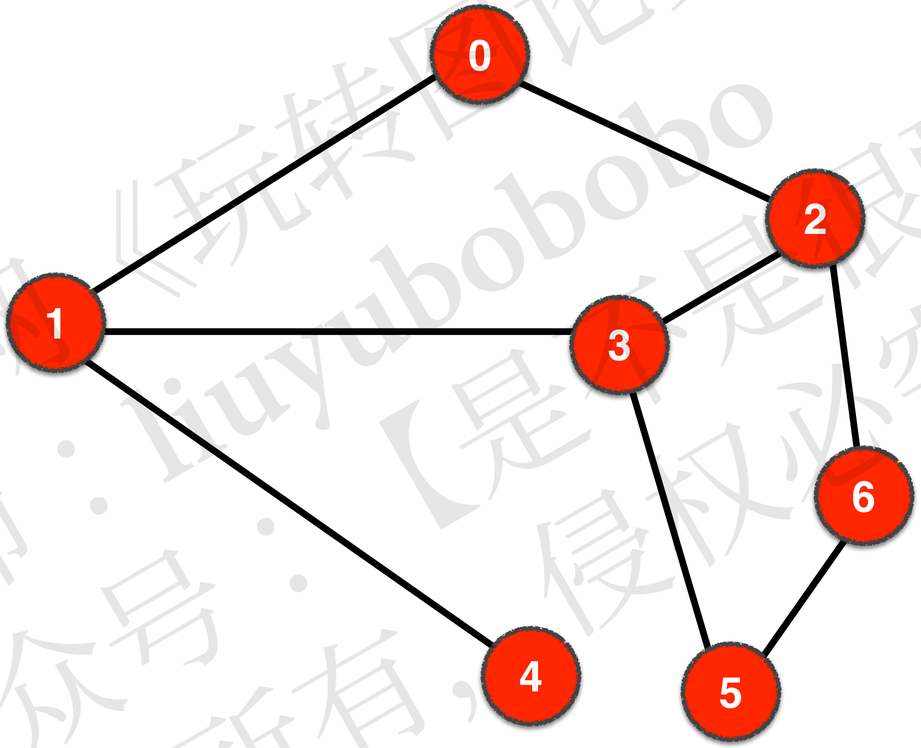
0	1	2	3	4	5	6

遍历结果：0 1 3 2 6 5 4

0:	1	2	
1:	0	3	4
2:	0	3	6
3:	1	2	5
4:	1		
5:	3	6	
6:	2	5	

图的深度优先遍历

```
visited[0…V-1] = false;
dfs(0);
dfs(int v){
      visited[v] = true;
      list.add(v);
      for(int w: adj(v))
          if(!visited[w])
              dfs(w);
}
dfs(0)
```



visited

0	1	2	3	4	5	6

遍历结果：0 1 3 2 6 5 4

0: 1 2

1: 0 3 4

2: 0 3 6

3: 1 2 5

4: 1

5: 3 6

6: 2 5

实现图的深度优先遍历

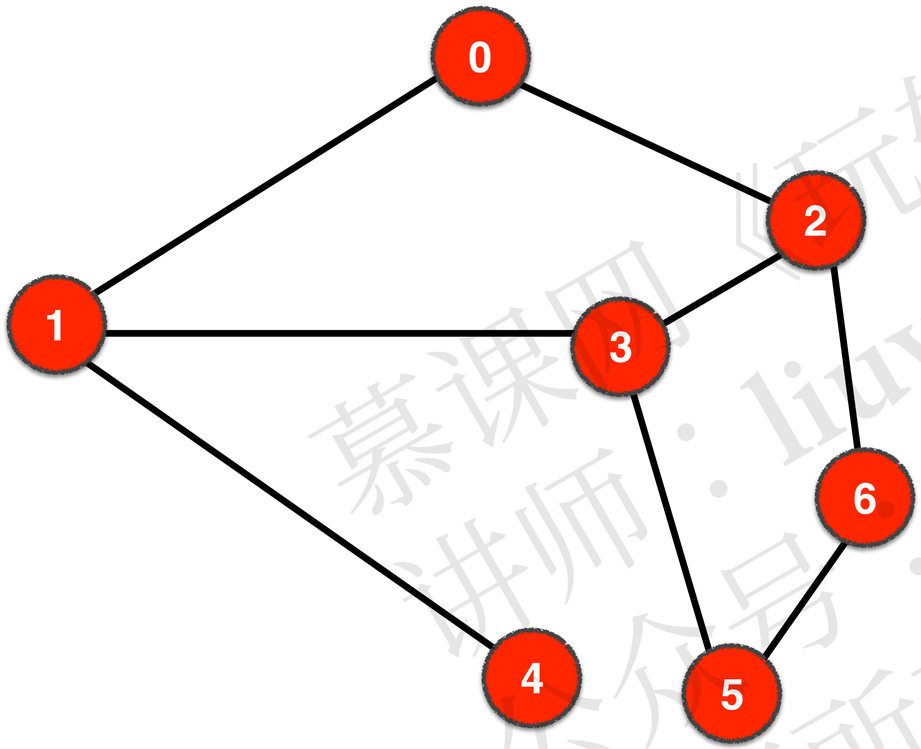
liuyubobobo

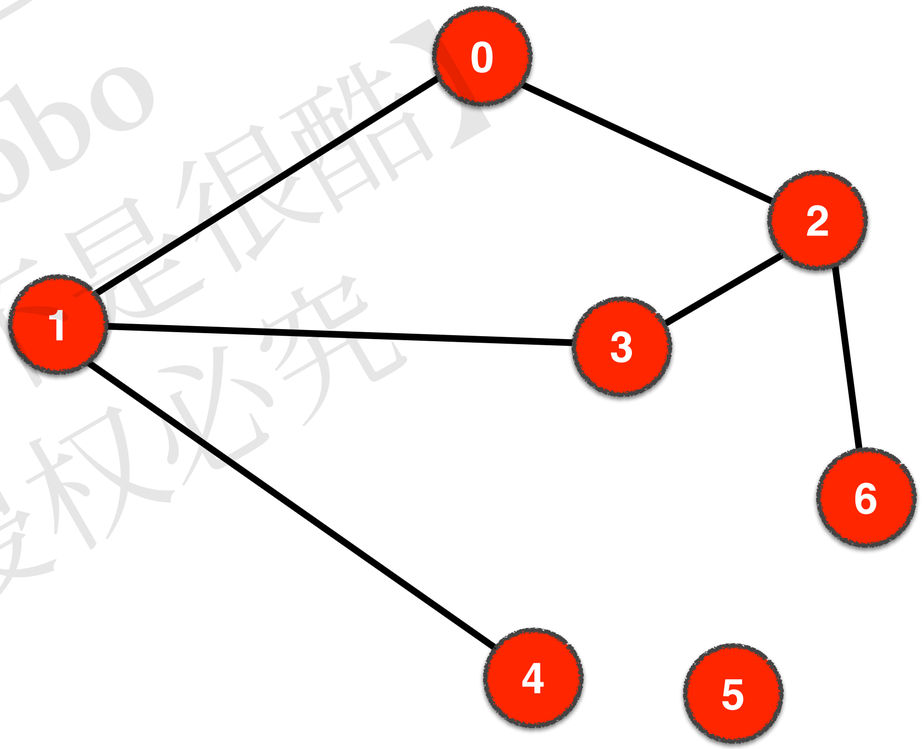
编程实践：实现图的深度优先遍历

图的深度优先遍历的改进

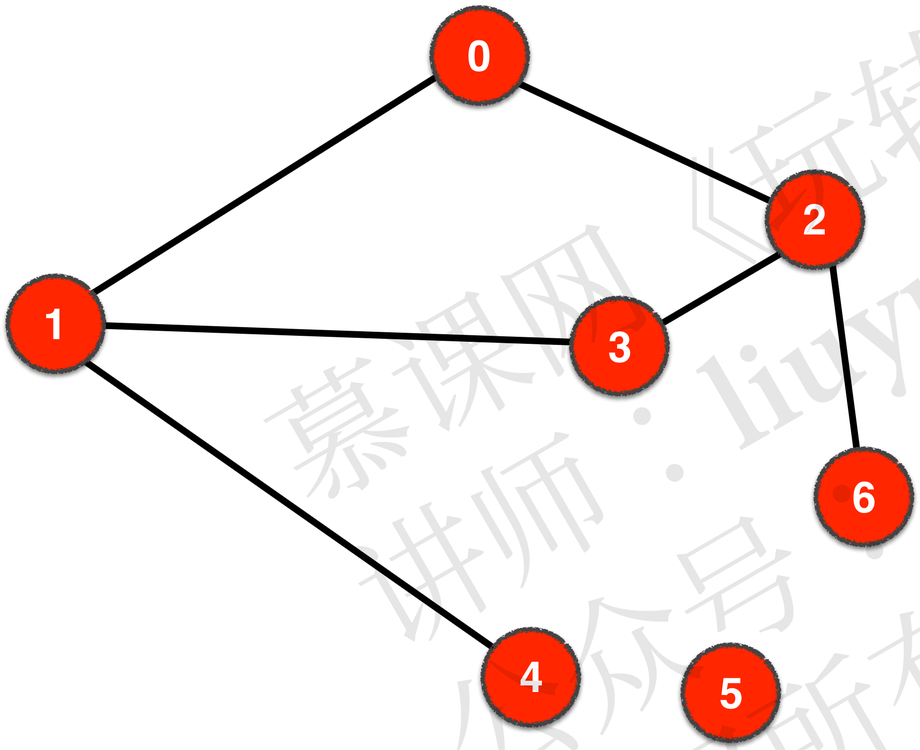
liuyubobobo

图的深度优先遍历的改进



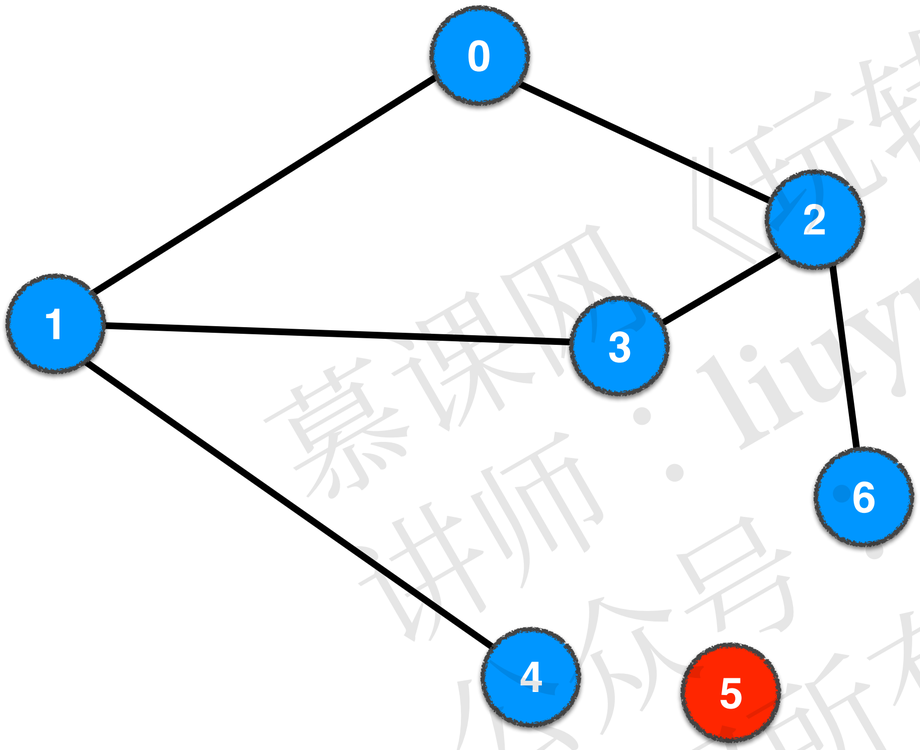


图的深度优先遍历的改进




```
visited[0…V] = false;
dfs(0);
dfs(int v){
        visited[w] = true;
        list.add(v);
        for(int w: adj(v))
            if(!visited[w])
                dfs(w);
}
```

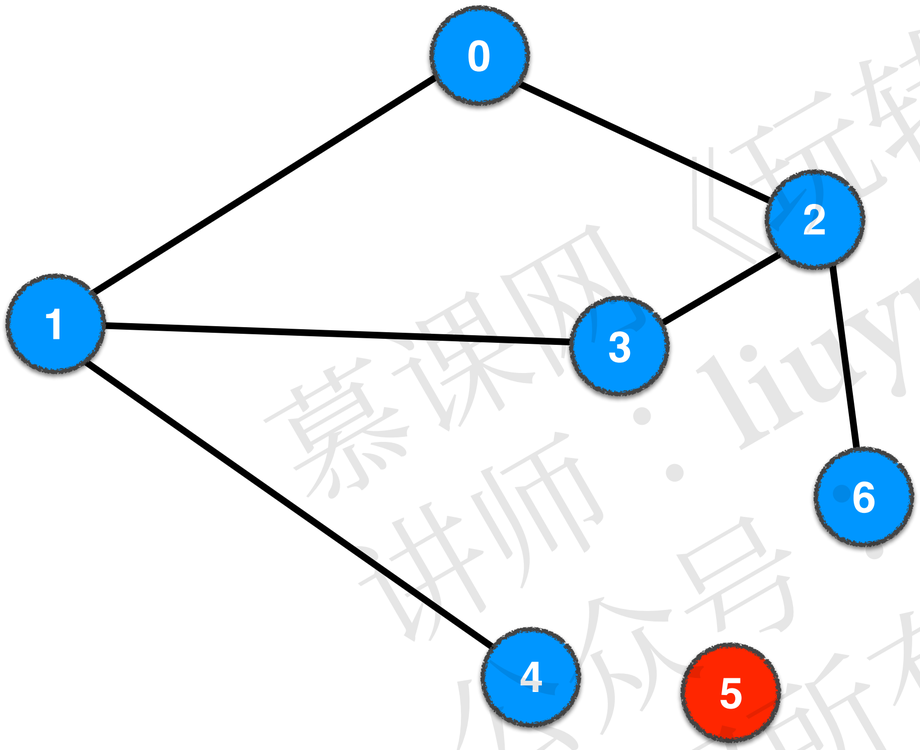
图的深度优先遍历的改进




```
visited[0…V] = false;
dfs(0);
dfs(int v){
        visited[w] = true;
        list.add(v);
        for(int w: adj(v))
            if(!visited[w])
                dfs(w);
}
```

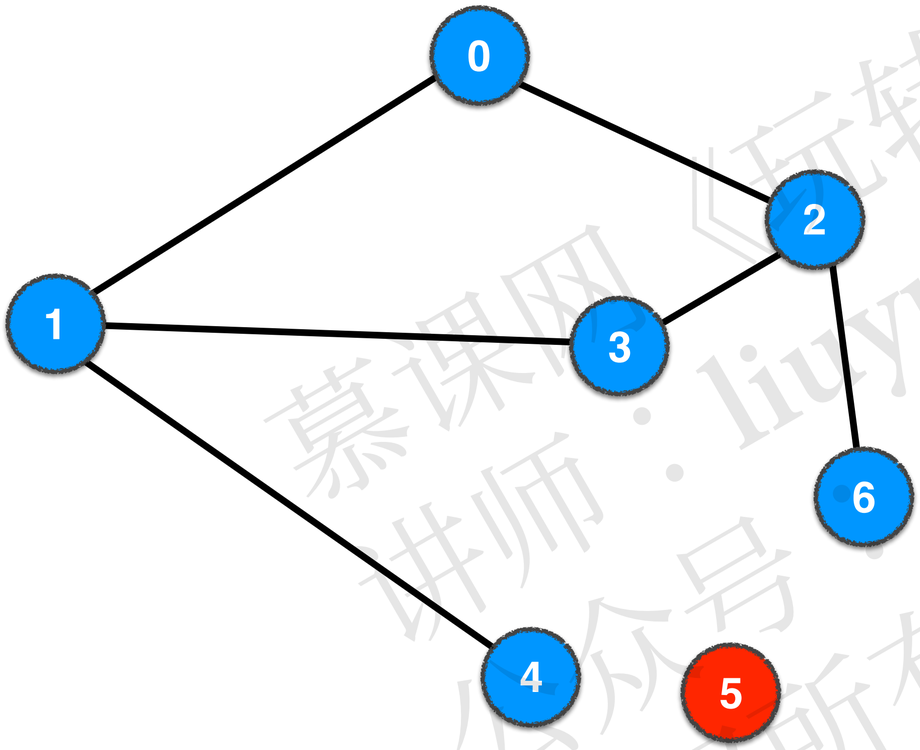
测试：现在dfs的bug

图的深度优先遍历的改进




```
visited[0…V] = false;
dfs(0);
dfs(int v){
        visited[w] = true;
        list.add(v);
        for(int w: adj(v))
            if(!visited[w])
                dfs(w);
}
```

图的深度优先遍历的改进




```
visited[0…V] = false;

for(int v = 0; v < V; v ++)
    if(!visited[v])
        dfs(v);

dfs(int v){

    visited[w] = true;

    list.add(v);

    for(int w: adj(v))

        if(!visited[w])

            dfs(w);

}
```

编程实践：改进图的深度优先遍历

回顾二叉树的遍历

```
preorder(TreeNode node){
    if(node != null){
        list.add(node.val);
        preorder(node.left);
        preorder(node.right);
    }
}
```

```
inorder(TreeNode node){
    if(node != null){
        inorder(node.left);
        list.add(node.val);
        inorder(node.right);
    }
}
```

```
postorder(TreeNode node){
    if(node != null){
        postorder(node.left);
        postorder(node.right);
        list.add(node.val);
    }
}
```

图的遍历

```
visited[0…V] = false;

for(int v = 0; v < V; v ++)
    if(!visited[v])
        dfs(v);

dfs(int v){

      visited[w] = true;

      list.add(v);

      for(int w: adj(v))

          if(!visited[w])

              dfs(w);

}
```

深度优先先序遍历

```
visited[0…V] = false;

for(int v = 0; v < V; v ++)
    if(!visited[v])
        dfs(v);

dfs(int v){

      visited[w] = true;

      for(int w: adj(v))

          if(!visited[w])

              dfs(w);

      list.add(v);

}
```

深度优先后序遍历

编程实践：深度优先序遍历

更多关于图的深度优先遍历

liuyubobobo

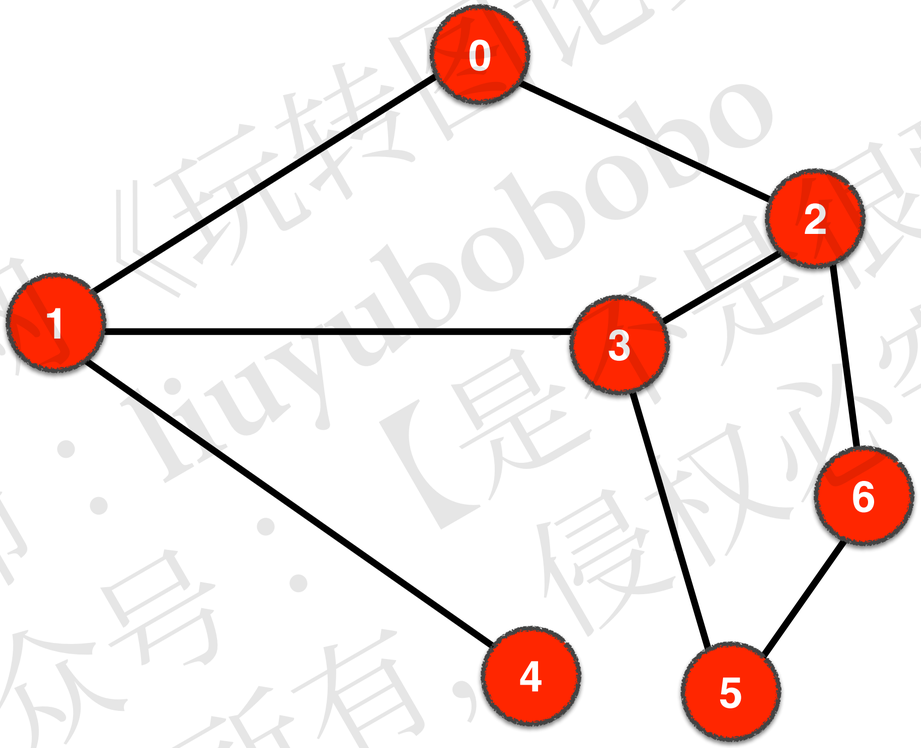
更多关于图的深度优先遍历

复杂度分析

图的深度优先遍历

```
visited[0…V-1] = false;

for(int v = 0; v < V; v ++)
    if(!visited[v])
        dfs(v);

dfs(int v){

      visited[v] = true;

      list.add(v);

      for(int w: adj(v))

          if(!visited[w])

              dfs(w);

}
```

复杂度：O（V + E）



visited

0	1	2	3	4	5	6

遍历结果：0 1 3 2 6 5 4

0: 1 2

1: 0 3 4

2: 0 3 6

3: 1 2 5

4: 1

5: 3 6

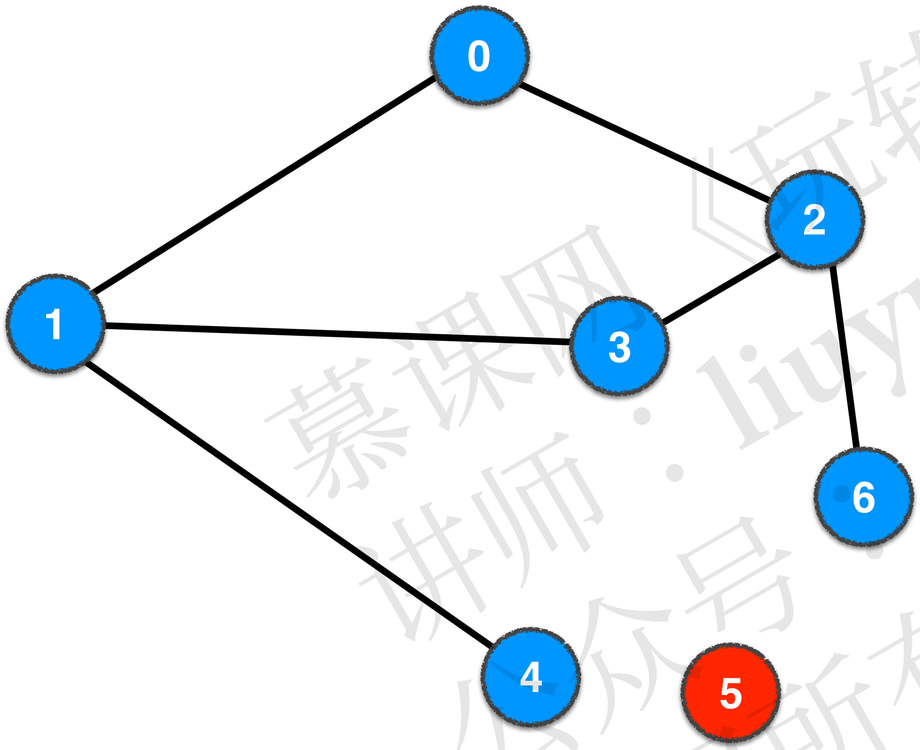
6: 2 5

更多关于图的深度优先遍历

图的深度优先遍历，复杂度：

O（V + E）

深度优先遍历的应用



求图的联通分量

求两点间是否可达

求两点间的一条路径

检测图中是否有环?

深度优先遍历的应用

二分图检测

寻找图中的桥

寻找图中的割点

哈密尔顿路径

拓扑排序

更多关于图的深度优先遍历

深度优先遍历的非递归实现

图的邻接矩阵的深度优先遍历

大家加油！

欢迎大家关注我的个人公众号：是不是很酷

坚持有质量的技术原创

用技术人的视角看世界

「是不是很酷」

文字内容：深度优先遍历的非递归写法

文字内容：邻接矩阵的深度优先遍历

玩儿转图论算法

liuyubobobo