

# 极客大学算法训练营

## 第三课

### 数组、链表、跳表

覃超

Sophon Tech 创始人，前 Facebook 工程师

# 目录

- 第一节 数组、链表、跳表基本实现和特性
- 第二节 实战题目解析

## 第一节

# 数组、链表、跳表的基本实现和特性

# Array

Java, C++:      `int a[100];`

Python:          `list = []`

JavaScript:      `let x = [1, 2, 3]`

# Array

0	00000011
1	00000001
2	00000101
3	00000100
4	00000011
5	00000000
6	00000000
7	00000000
8	00000000

Memory  
Controller



00000000
00000001
00000011
00000011
00000100
00000100
00000101
00000110
00000110
00000111
00001000
00001010
00001010

# Array 增加元素

Inserting

0	A
1	B
2	C
3	E
4	F
5	G
6	

# Array 增加元素

Inserting

0	A	
1	B	
2	C	
3	E	D
4	F	
5	G	
6		

# Array 增加元素

Inserting

0	A	
1	B	
2	C	
3		← D
4	E	
5	F	
6	G	



# Array 增加元素

Inserting

0	A
1	B
2	C
3	D
4	E
5	F
6	G

# Array 删除元素


Deleting

0	A
1	B
2	C
3	Z
4	D
5	E
6	F

# Array 删除元素

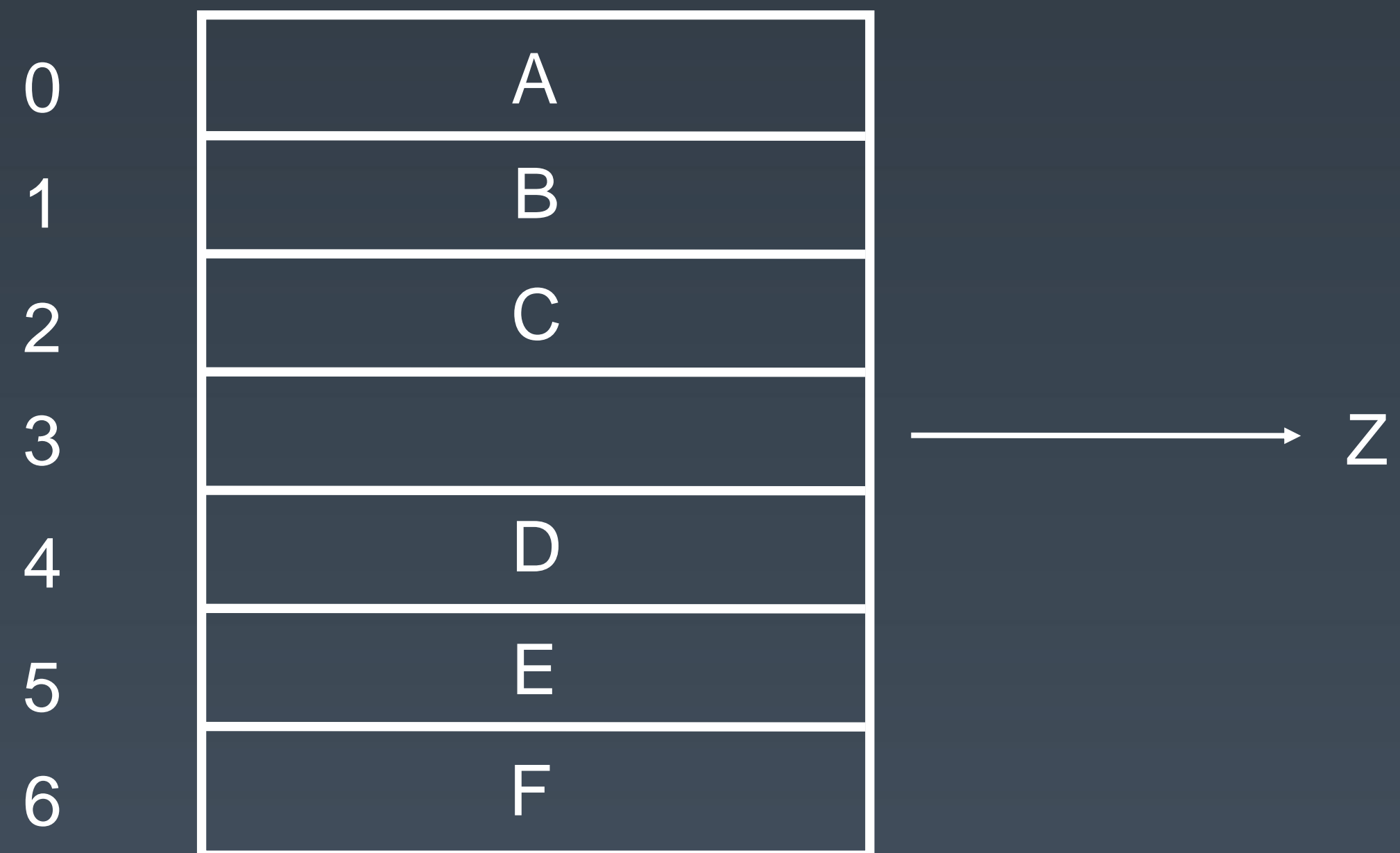
Deleting

0	A
1	B
2	C
3	Z
4	D
5	E
6	F



# Array 删除元素

Deleting

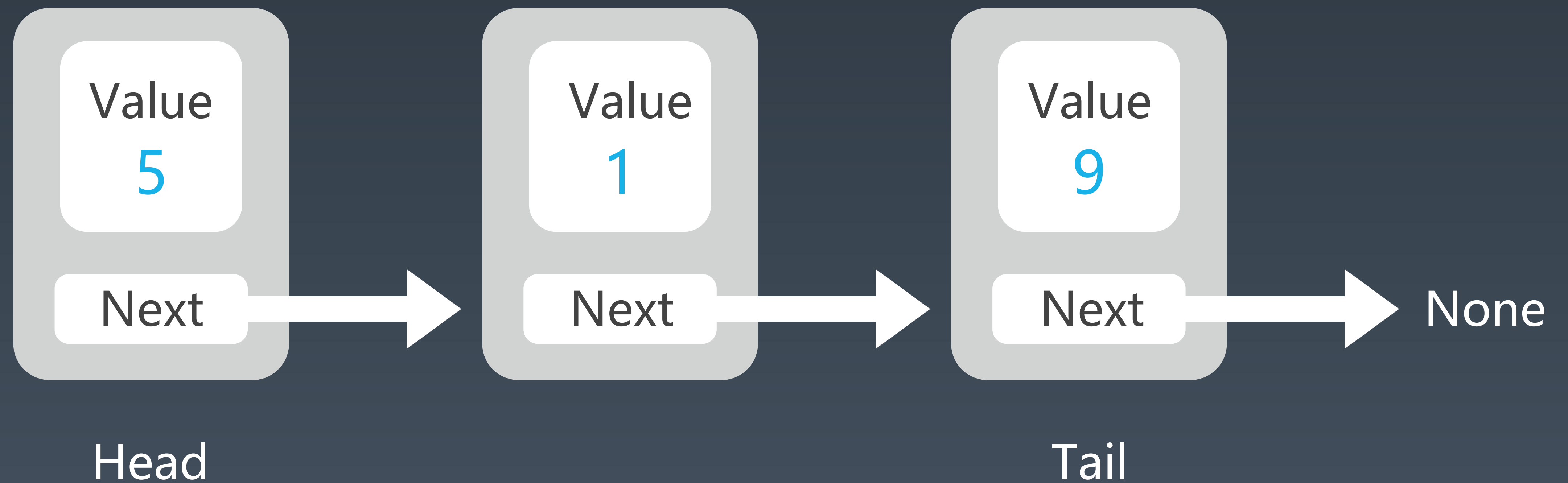


# Array 删除元素

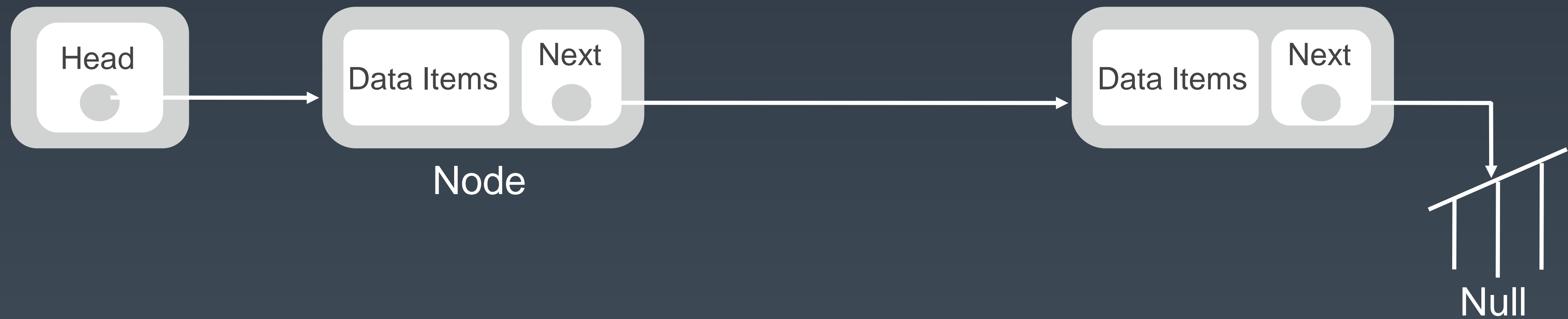
Deleting

0	A	Z
1	B	
2	C	
3	D	
4	E	
5	F	
6		

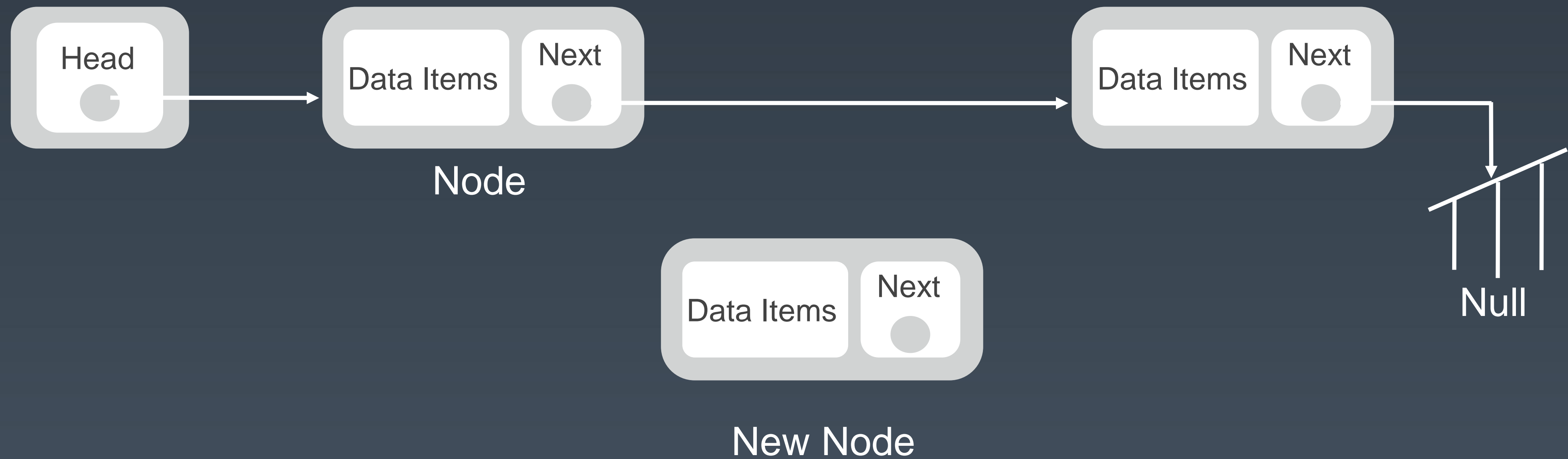
# Linked List



# Linked List 增加结点

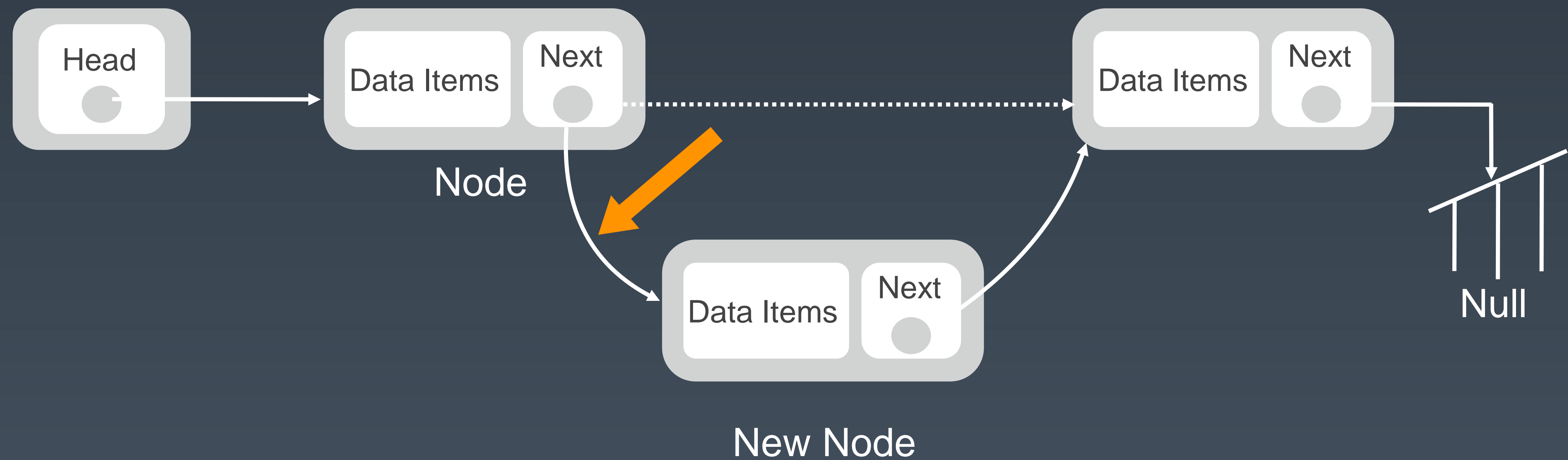


# Linked List 增加结点

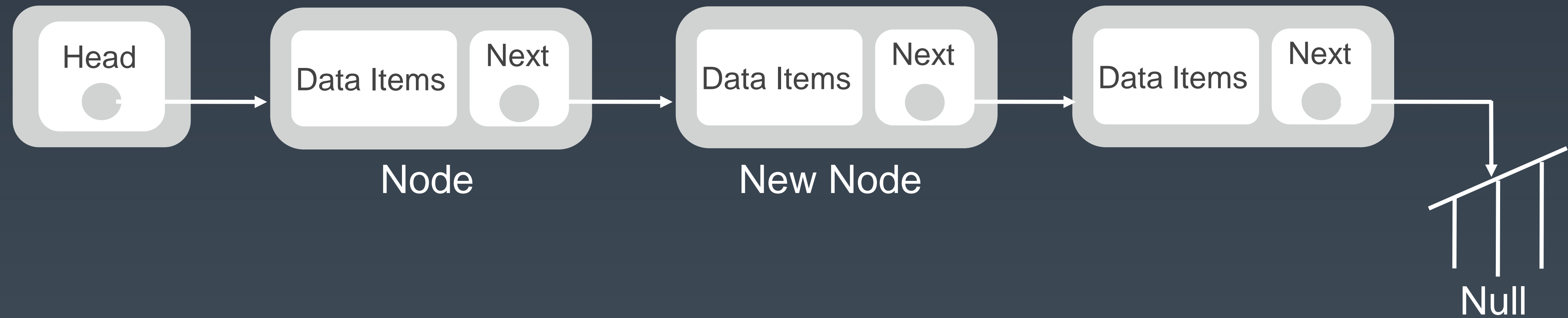




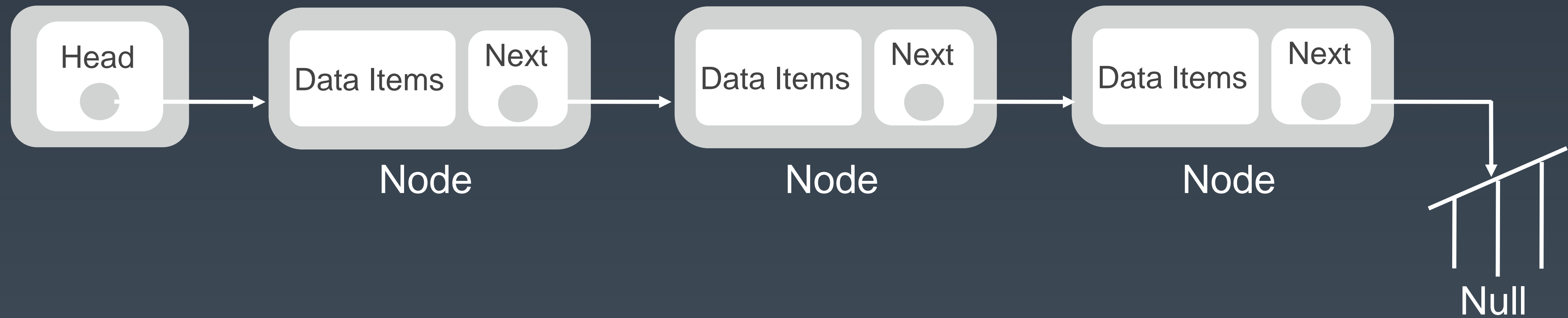
# Linked List 增加结点



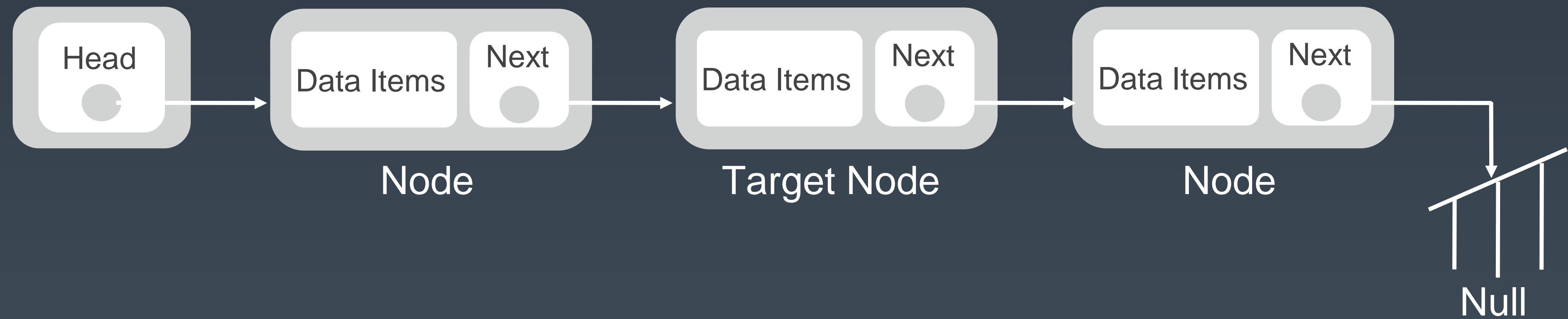
# Linked List 增加结点



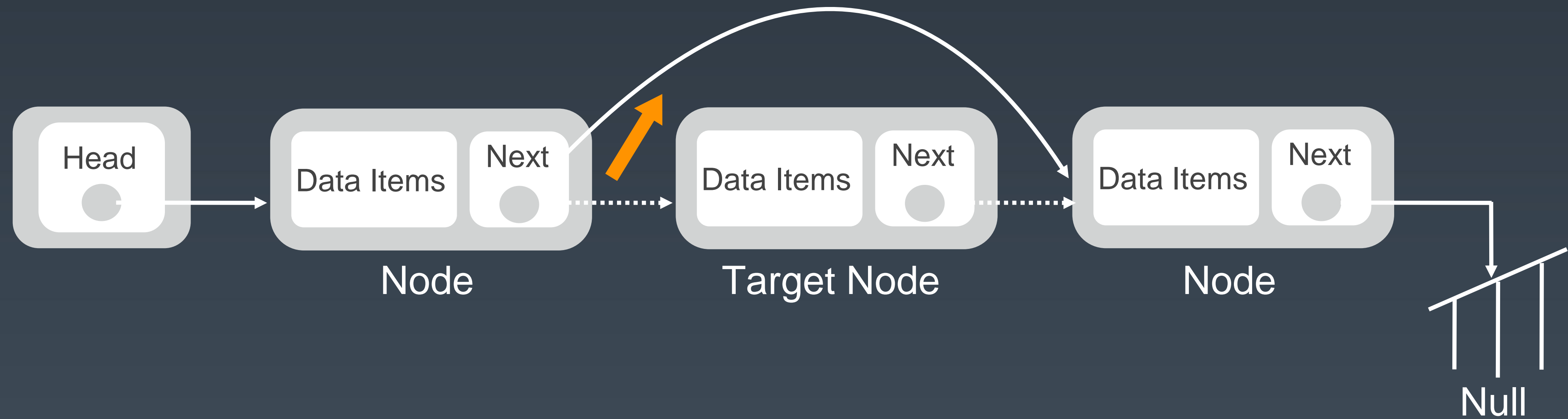
# Linked List 删除结点



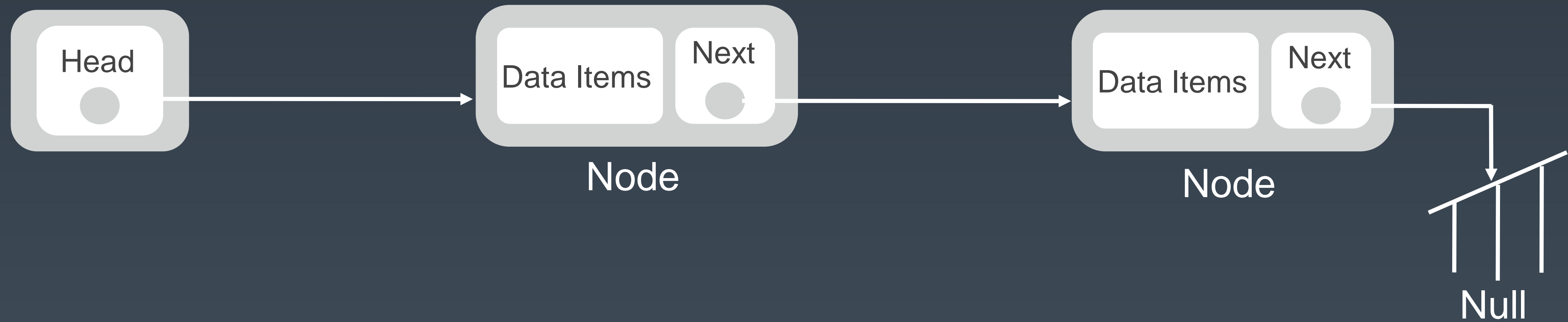
# Linked List 删除结点



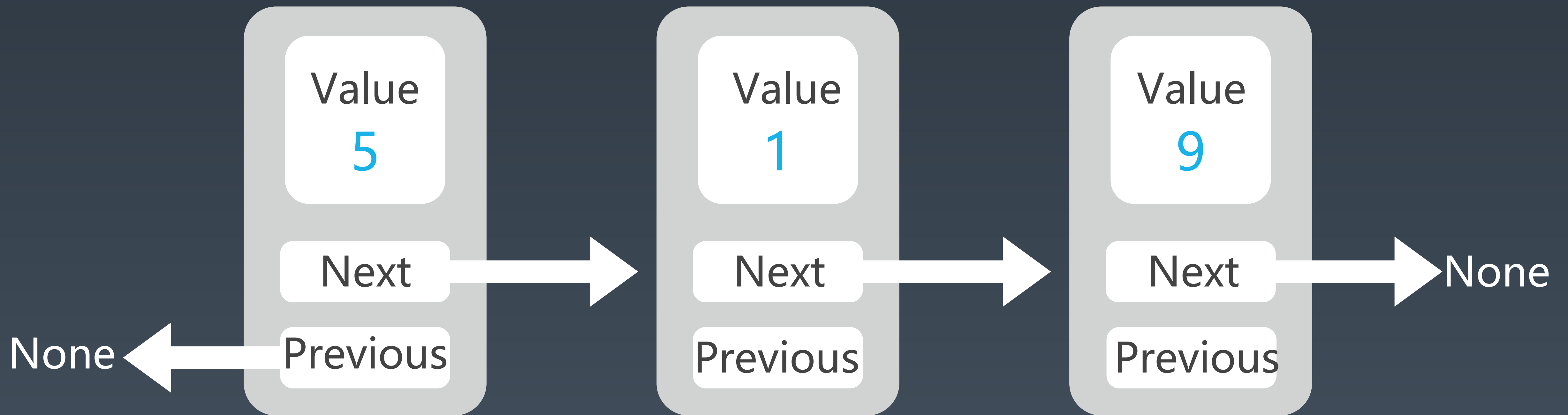
# Linked List 删除结点



# Linked List 删除结点



# Double Linked List



# 时间复杂度

prepend  $O(1)$

append  $O(1)$

lookup  $O(n)$

insert  $O(1)$

delete  $O(1)$

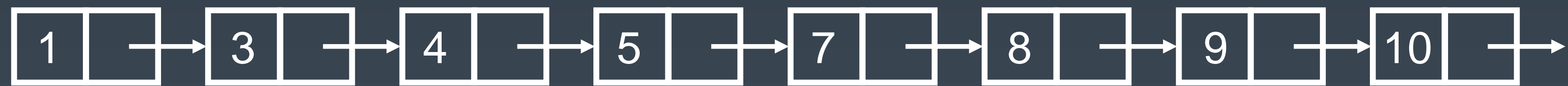


# 跳表 Skip List

# 普通链表

prepend	$O(1)$
append	$O(1)$
lookup	$O(n)$
insert	$O(1)$
delete	$O(1)$

# 链表元素有序的时候

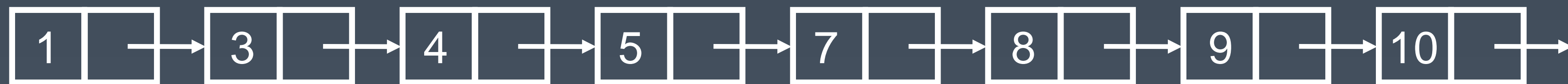


# 跳表的特点

**注意：只能用于元素有序的情况。**

所以，跳表 (skip list) 对标的是平衡树 (AVL Tree) 和二分查找，是一种 插入/删除/搜索 都是  $O(\log n)$  的数据结构。1989 年出现。

它最大的优势是原理简单、容易实现、方便扩展、效率更高。因此在一些热门的项目里用来替代平衡树，如 Redis、LevelDB 等。



# 如何给有序的链表加速



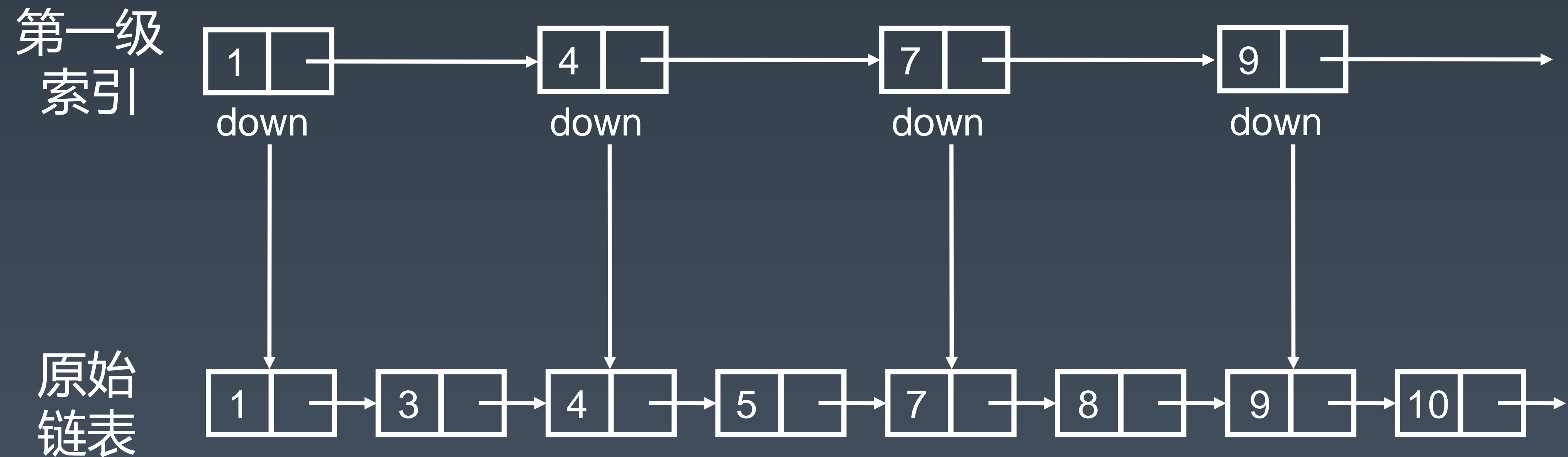
时间复杂度：查询  $O(n)$

简单优化：添加头尾指针

然后呢？ — 思考

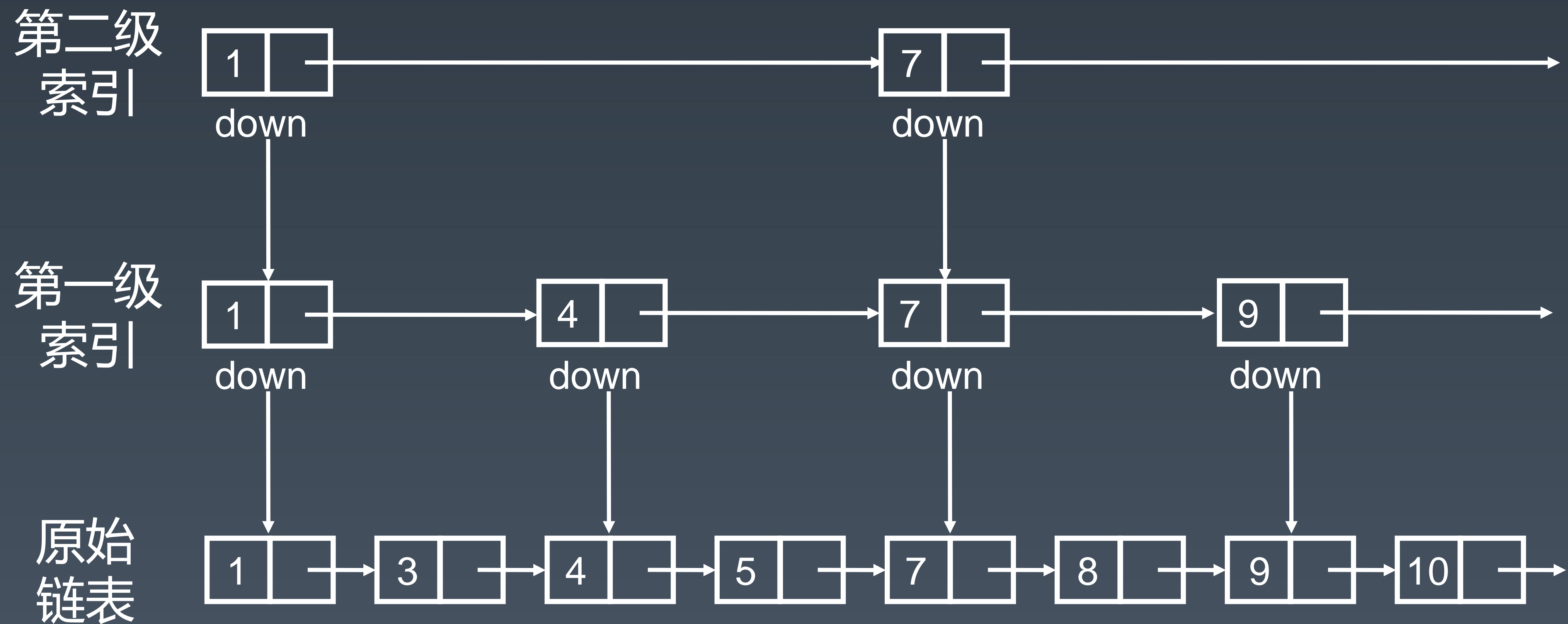
# 添加第一级索引

如何提高链表线性查找的效率？



# 添加第二级索引

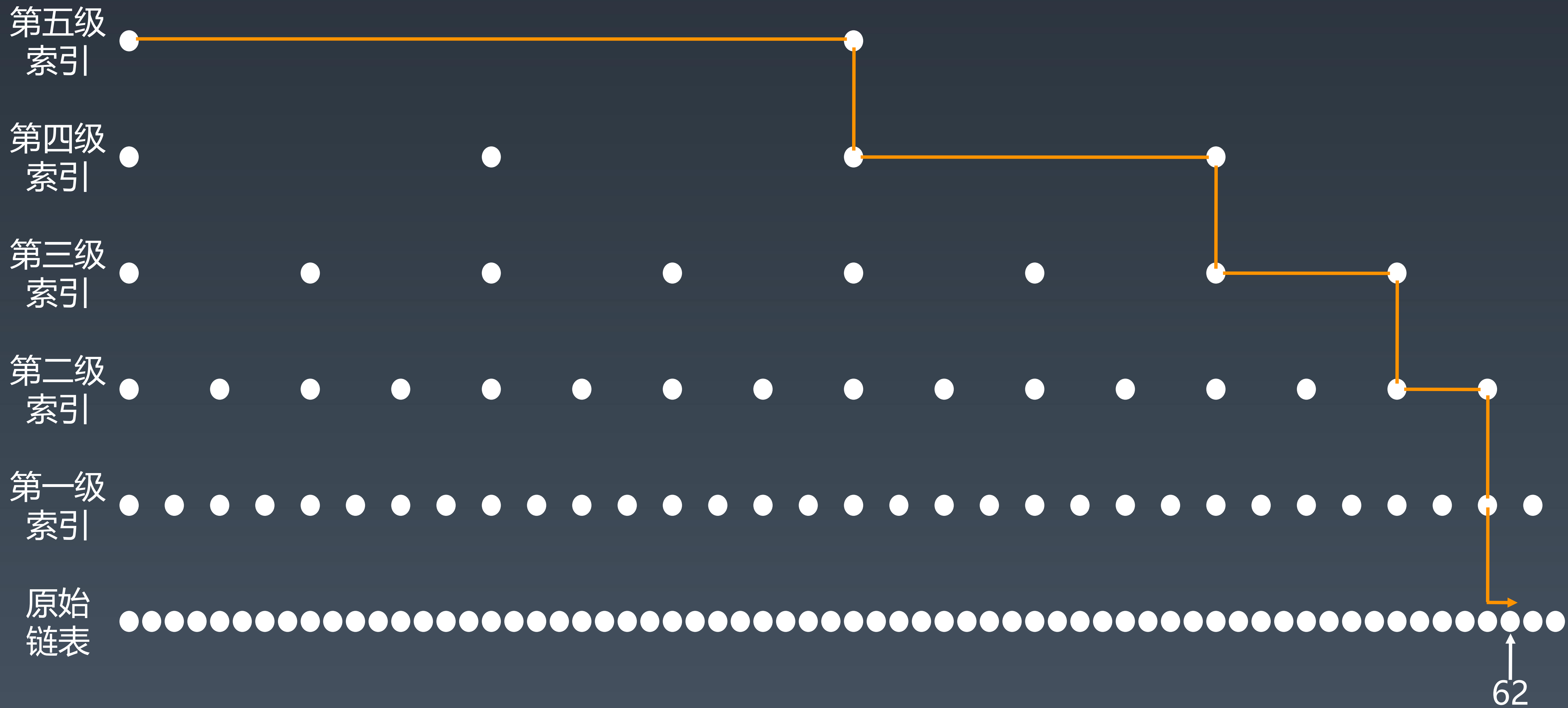
如何进一步提高链表查找的效率？



# 添加多级索引

以此类推，增加多级索引



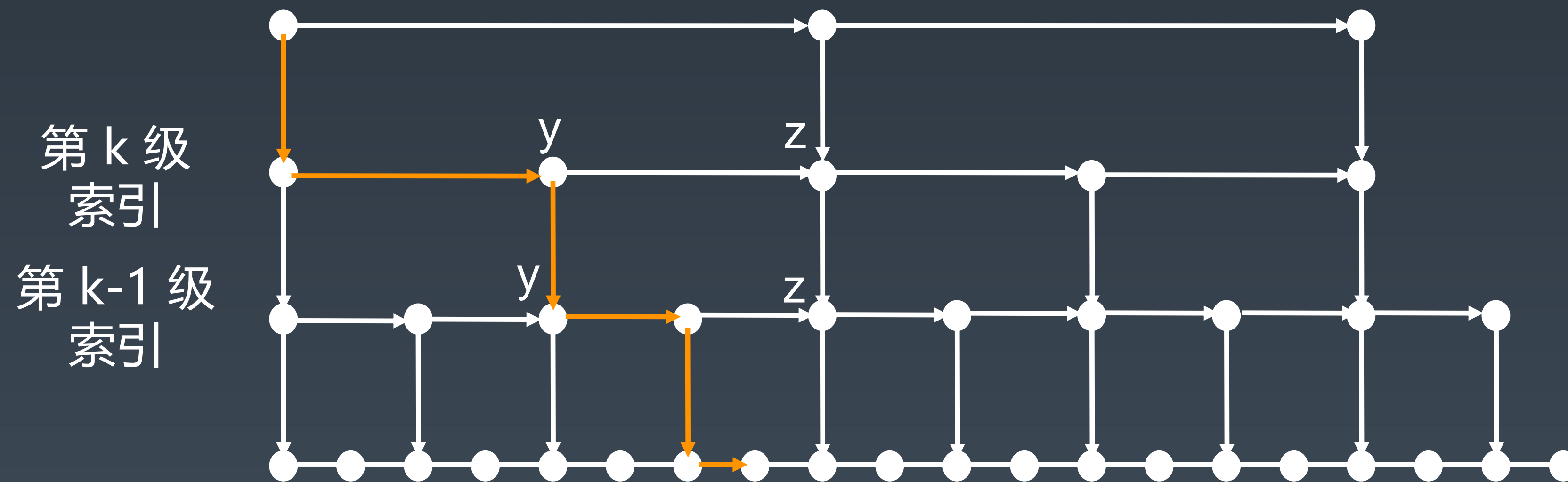


# 跳表查询的时间复杂度分析

$n/2$ 、 $n/4$ 、 $n/8$ 、第  $k$  级索引结点的个数就是  $n/(2^k)$

假设索引有  $h$  级，最高级的索引有 2 个结点。 $n/(2^h) = 2$ ，从而求得  $h = \log_2(n)-1$

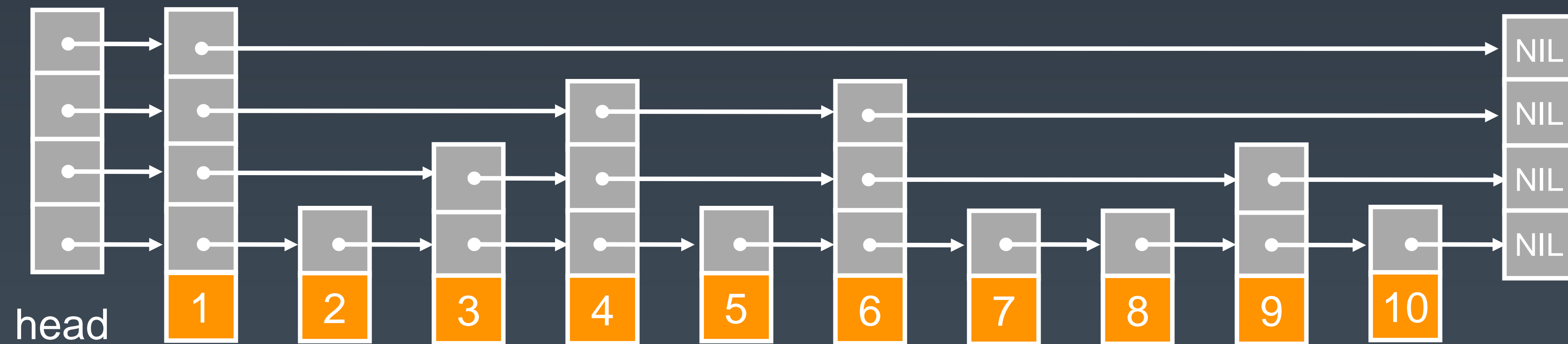
# 跳表查询的时间复杂度分析



索引的高度:  $\log n$ , 每层索引遍历的结点个数: 3

在跳表中查询任意数据的时间复杂度就是  $O(\log n)$

# 现实中跳表的形态



# 跳表的空间复杂度分析

原始链表大小为  $n$ ，每 2 个结点抽 1 个，每层索引的结点数：

$$\frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \dots, 8, 4, 2$$

原始链表大小为  $n$ ，每 3 个结点抽 1 个，每层索引的结点数：

$$\frac{n}{3}, \frac{n}{9}, \frac{n}{27}, \dots, 9, 3, 1$$

空间复杂度是  $O(n)$

# 工程中的应用

LRU Cache - Linked list

<https://www.jianshu.com/p/b1ab4a170c3c>

<https://leetcode-cn.com/problems/lru-cache>

Redis - Skip List

<https://redisbook.readthedocs.io/en/latest/internal-datastruct/skiplist.html>

<https://www.zhihu.com/question/20202931>

# 小结

- 数组、链表、跳表的原理和实现
- 三者的时间复杂度、空间复杂度
- 工程运用
- 跳表：升维思想 + 空间换时间

## 第二节

# 实战题目解析



# 练习步骤

1. 5-10分钟：读题和思考
2. 有思路：自己开始做和写代码；不然，马上看题解！
3. 默写背诵、熟练
4. 然后开始自己写（闭卷）

# 实战练习题目 - Array

1. <https://leetcode-cn.com/problems/container-with-most-water/>
2. <https://leetcode-cn.com/problems/move-zeroes/>
3. <https://leetcode-cn.com/problems/climbing-stairs/>
4. <https://leetcode-cn.com/problems/3sum/> (高频老题)

# 实战练习题目 - Linked List

1. <https://leetcode-cn.com/problems/reverse-linked-list/>
2. <https://leetcode-cn.com/problems/swap-nodes-in-pairs>
3. [\*\*https://leetcode-cn.com/problems/linked-list-cycle\*\*](https://leetcode-cn.com/problems/linked-list-cycle)
4. <https://leetcode-cn.com/problems/linked-list-cycle-ii>
5. <https://leetcode-cn.com/problems/reverse-nodes-in-k-group/>

解法固定，熟能生巧

# Homework

1. <https://leetcode-cn.com/problems/remove-duplicates-from-sorted-array/>
2. <https://leetcode-cn.com/problems/rotate-array/>
3. <https://leetcode-cn.com/problems/merge-two-sorted-lists/>
4. <https://leetcode-cn.com/problems/merge-sorted-array/>
5. <https://leetcode-cn.com/problems/two-sum/>
6. <https://leetcode-cn.com/problems/move-zeroes/>
7. <https://leetcode-cn.com/problems/plus-one/>

THANKS! |  极客大学