Sort List

LeetCode 148.Sort List

Sort a linked list in $O(n \log n)$ time using constant space complexity.

Example 1:

Input: 4->2->1->3

Output: 1->2->3->4

```python
# Author: kilien
# 思路：归并排序，分解链表，化整为零，再自底向上合并排序
# time: O (nlogn)  space: O (nloagn)
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution(object):
    def merge(self, h1, h2):
        dummy = tail = ListNode(None)
        while h1 and h2:
            if h1.val < h2.val:
                tail.next, tail, h1 = h1, h1, h1.next
            else:
                tail.next, tail, h2 = h2, h2, h2.next

        tail.next = h1 or h2
        return dummy.next

    def sortList(self, head):
        if not head or not head.next:
            return head

        pre, slow, fast = None, head, head
        while fast and fast.next:
            pre, slow, fast = slow, slow.next, fast.next.next
```

```
        pre.next = None

        return self.merge(*map(self.sortList, (head, slow)))
```

map 运用：
map() 函数接收两个参数，一个是函数，一个是Iterable，map将传入的函数依次作用到序列的每个元素，并把结果作为新的Iterator返回。

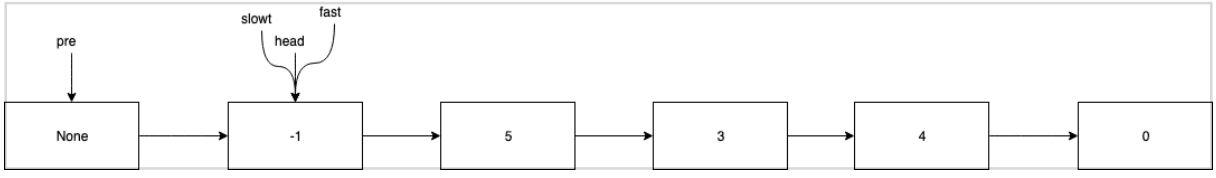
```
>>> def f(x):
...     return x * x
...
>>> r = map(f, [1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> list(r)
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

map()传入的第一个参数是f，即函数对象本身。由于结果r是一个Iterator，Iterator是惰性序列，因此通过list()函数让它把整个序列都计算出来并返回一个list。

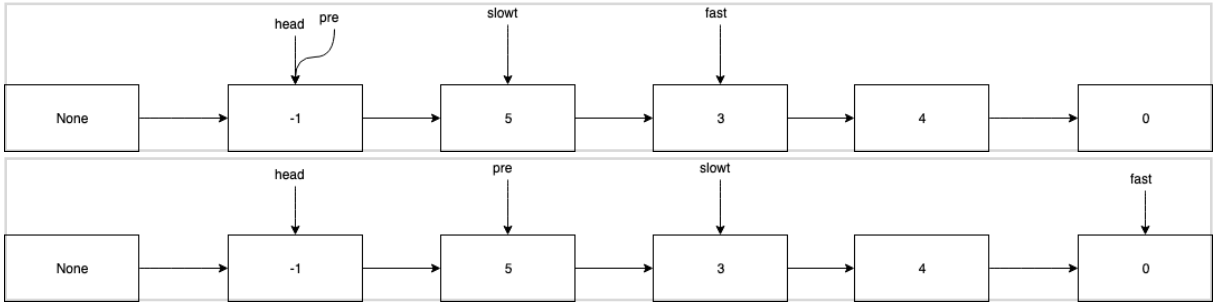
※ 的作用是将map对象作为实参传入merge函数。

具体流程可见图解：

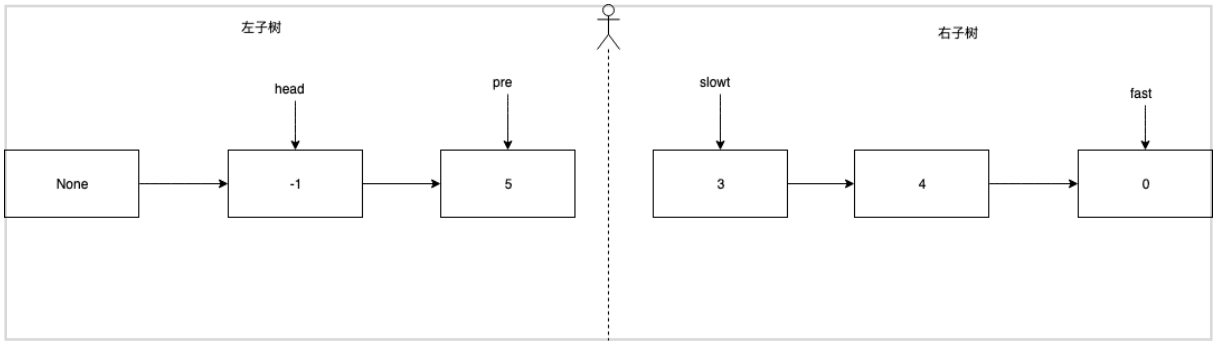
- 初始状态：



- sortlist循环：

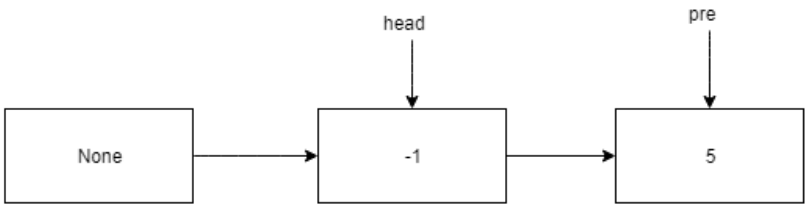


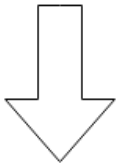
- 断链：pre.next = None



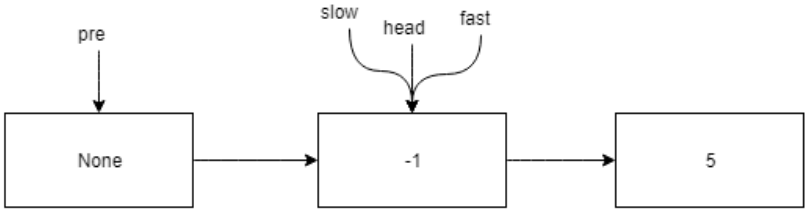
- 左子树拆分元素，合并排序

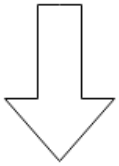
左子树

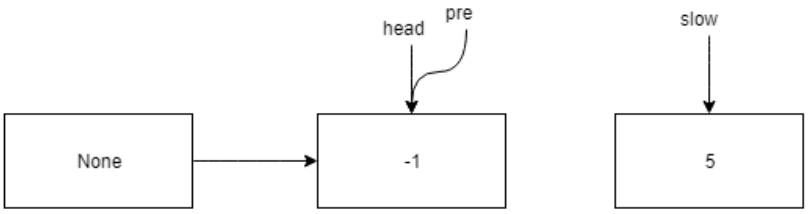

head
pre
None
-1
5

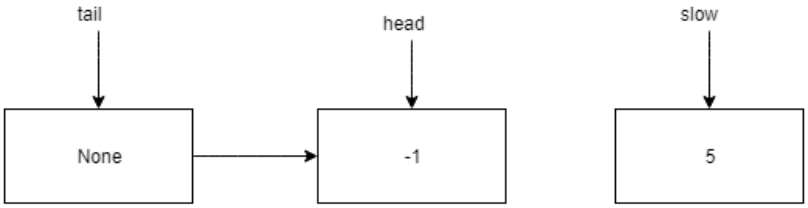


分解

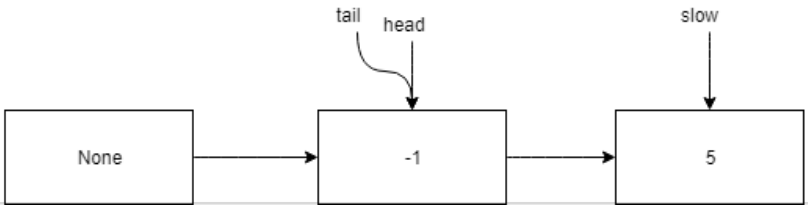

slow
head
fast
pre
None
-1
5



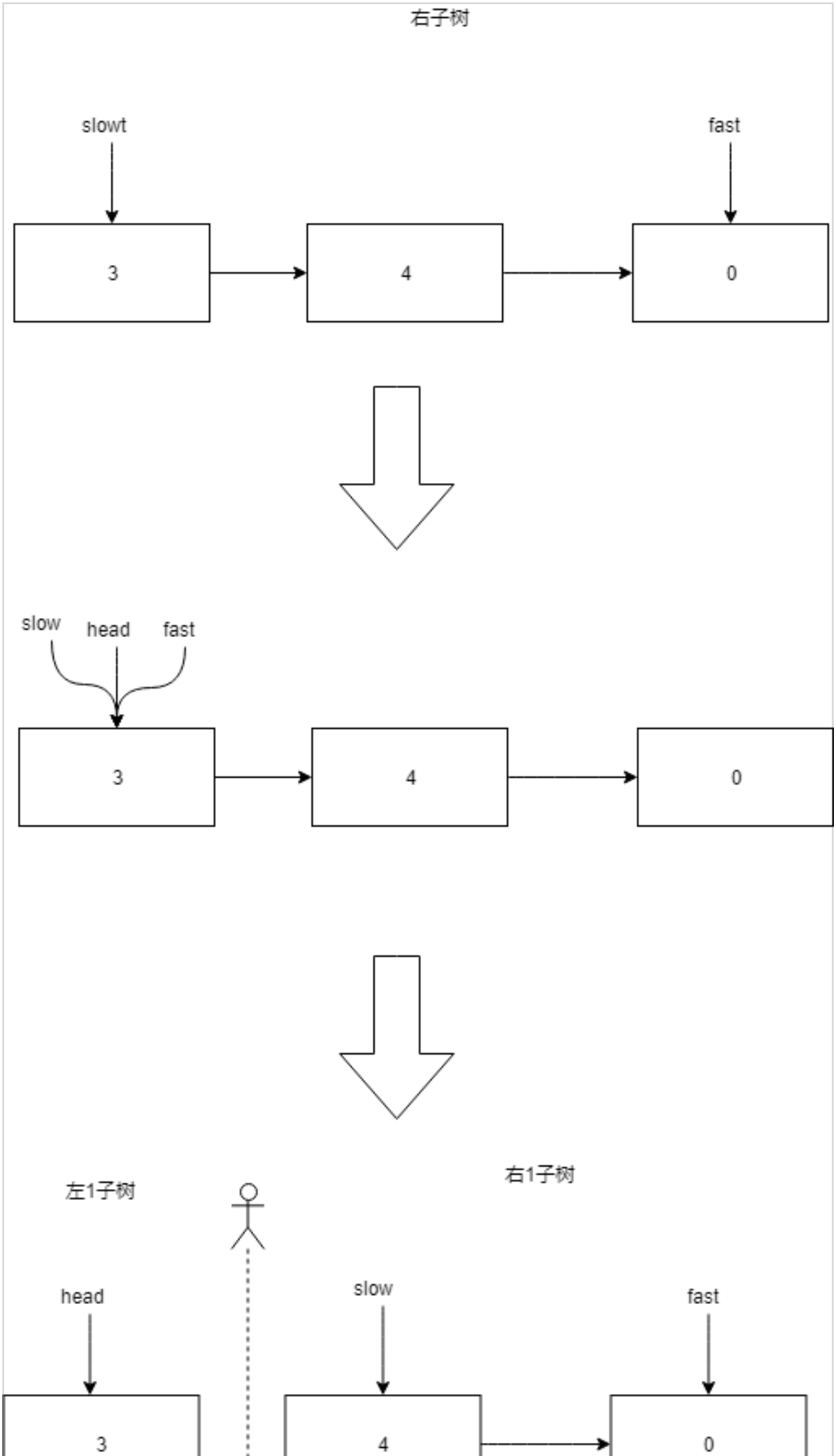

head
pre
slow
None
-1
5


tail
head
slow
None
-1
5

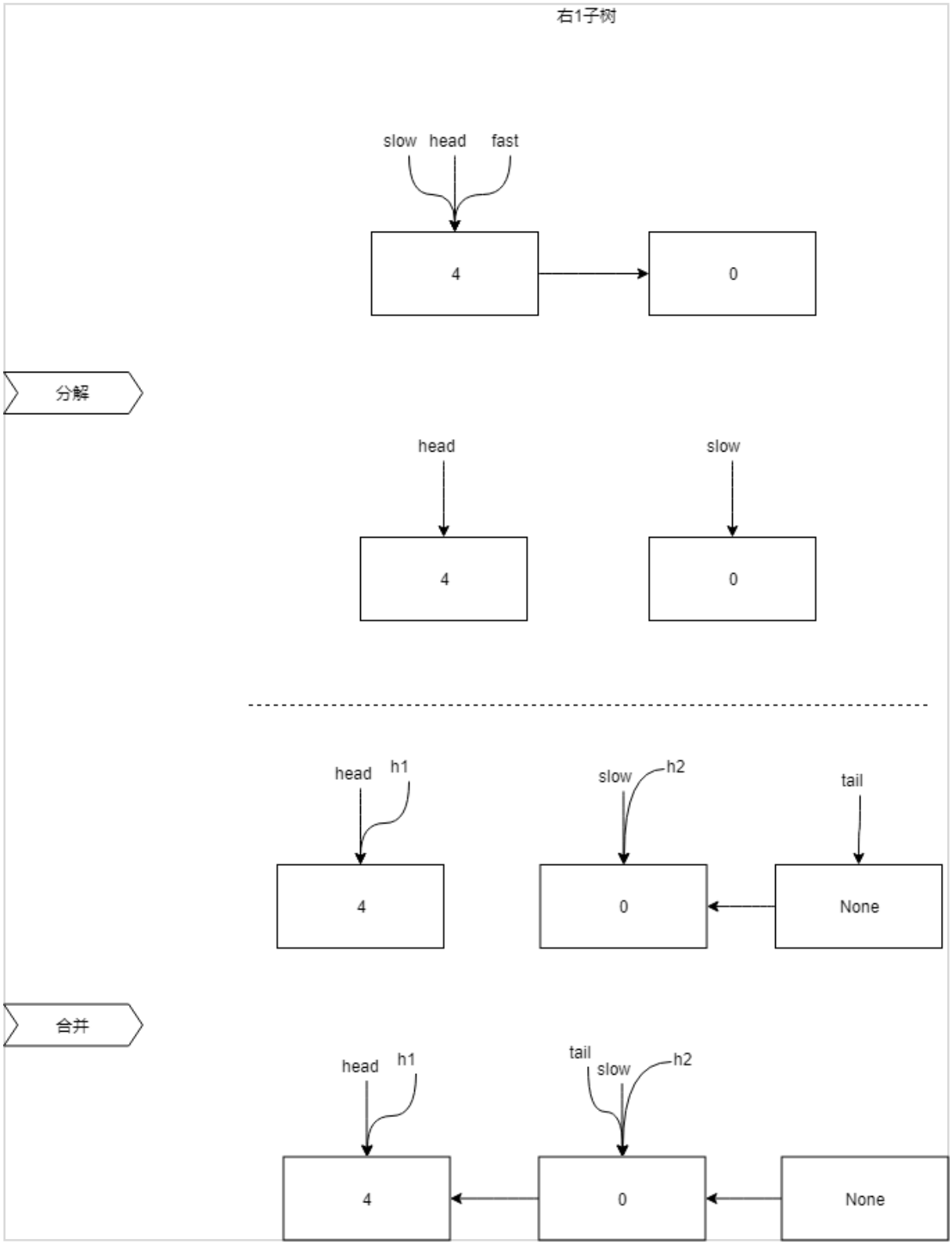
合并


tail
head
slow
None
-1
5

- 右子树拆分：左1子树和右1子树

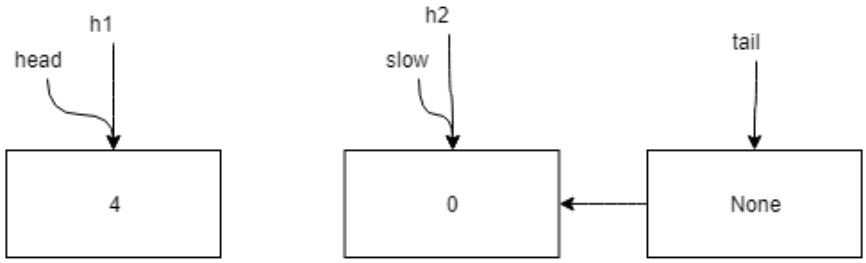

右子树
slowt
fast
3
4
0
slow
head
fast
3
4
0
左1子树
右1子树
head
slow
fast
3
4
0

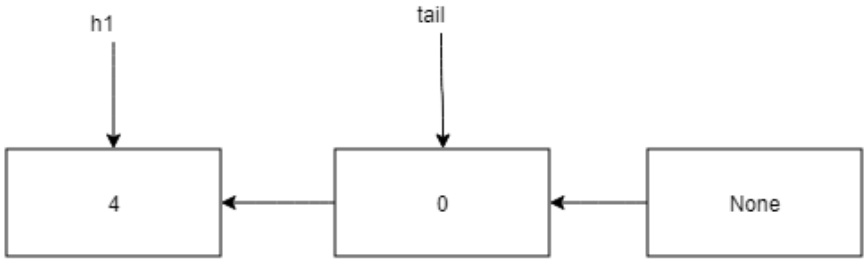
- 左1子树不变，右1子树拆分元素，合并

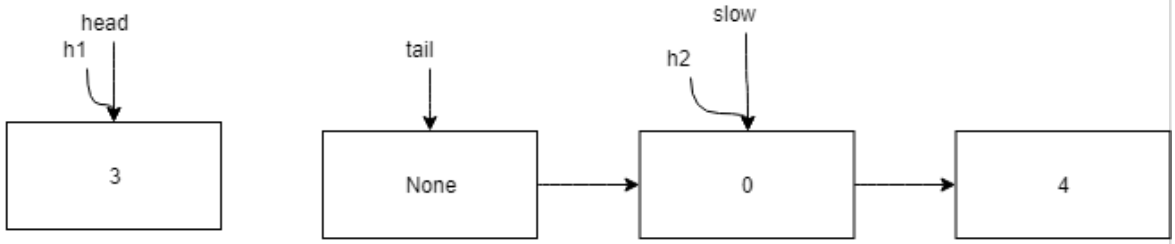


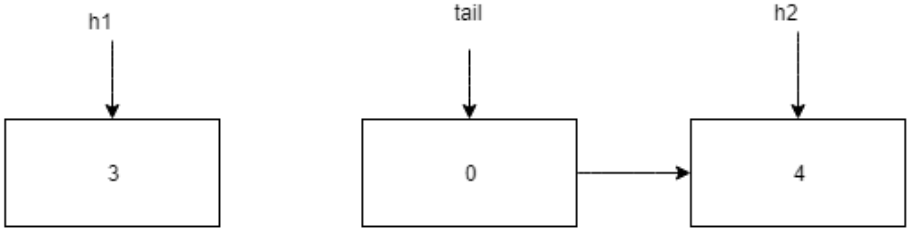
- 左1子树、右1子树合并

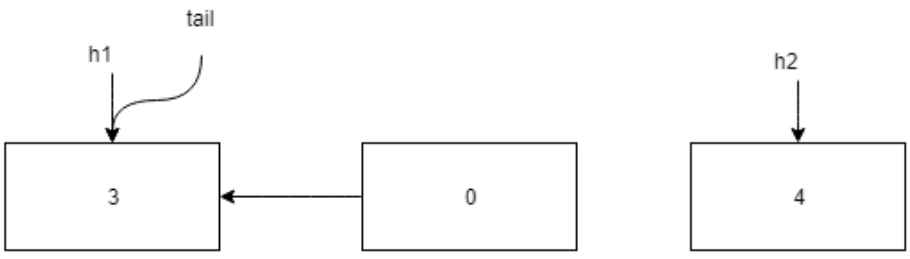
右子树


h1
head
h2
slow
tail
4
0
None

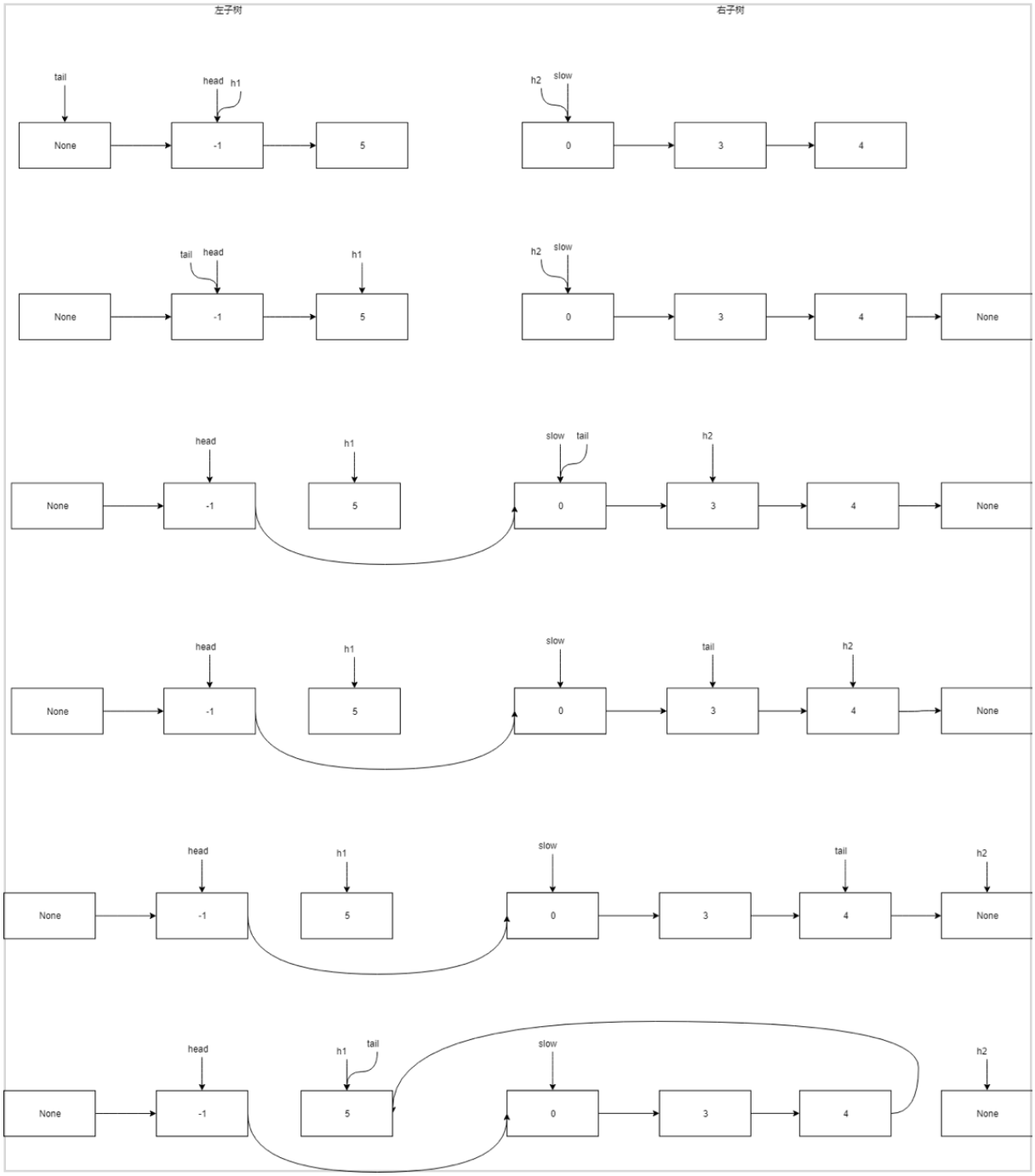

h1
tail
4
0
None


head
h1
tail
slow
h2
3
None
0
4


h1
tail
h2
3
0
4


tail
h1
h2
3
0
4

- 左右子树合并



#algorithm#