线程池与任务队列

胡船长

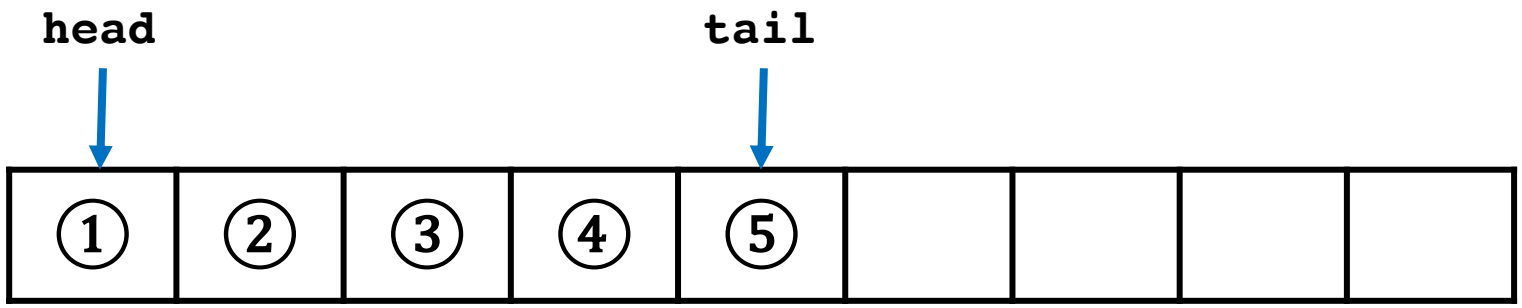
初航我带你，远航靠自己

队列基础知识

大约用时：（ 25 mins ）

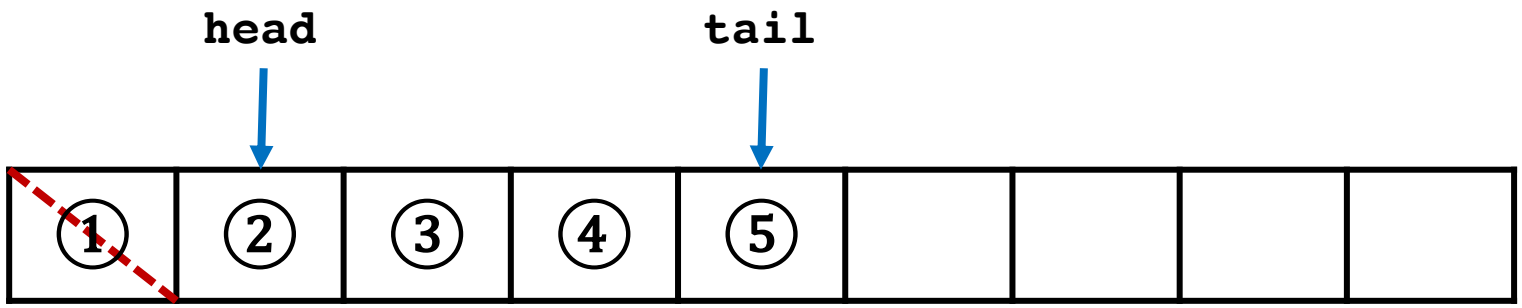
下一部分：队列的典型应用场景

队列



1、length = 9
2、head = 0
3、tail = 4
4、data_type = xxx

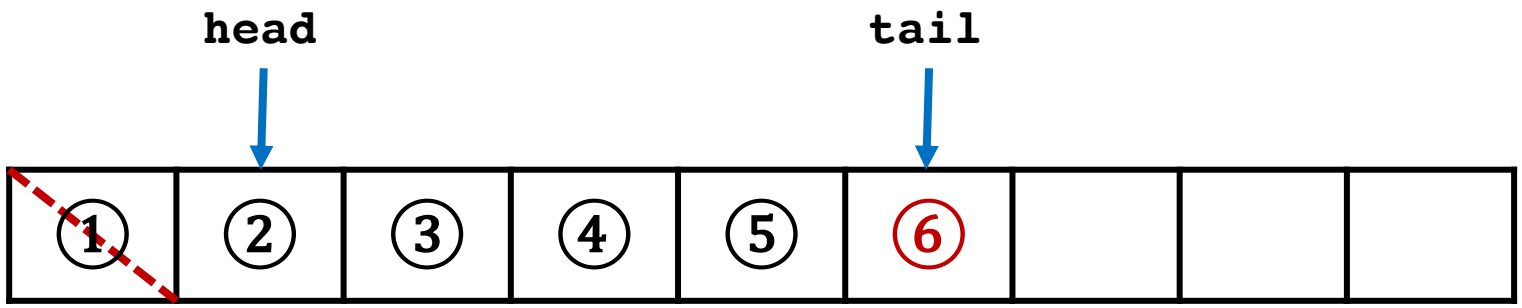
队列-出队



1、length = 9
2、head = 0
3、tail = 4
4、data_type = xxx

1、length = 9
2、head = 1
3、tail = 4
4、data_type = xxx

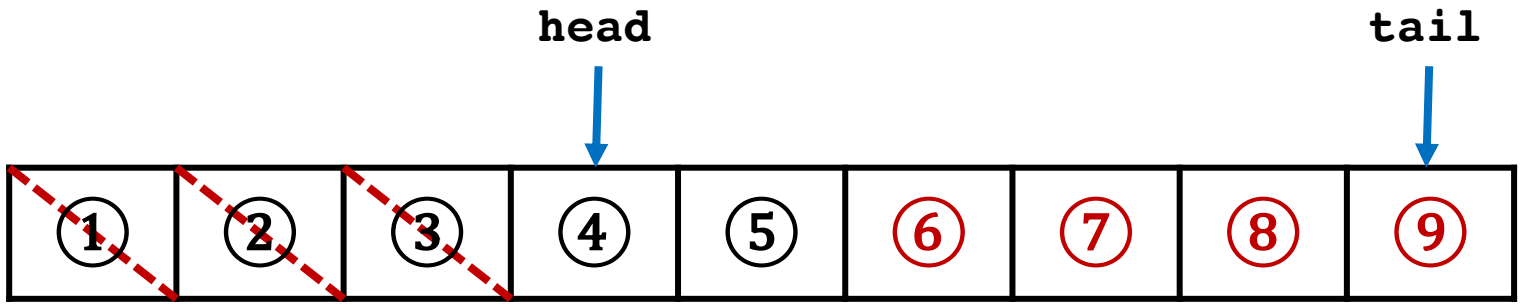
队列-入队



1、length = 9
2、head = 1
3、tail = 4
4、data_type = xxx

1、length = 9
2、head = 1
3、tail = 5
4、data_type = xxx

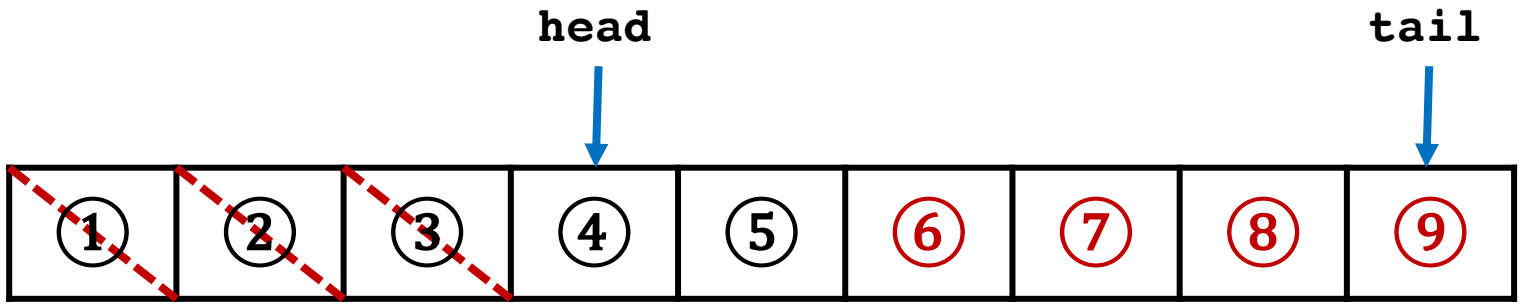
队列-假溢出



插入 ⑩ ？

1、length = 9
2、head = 3
3、tail = 8
4、data_type = xxx

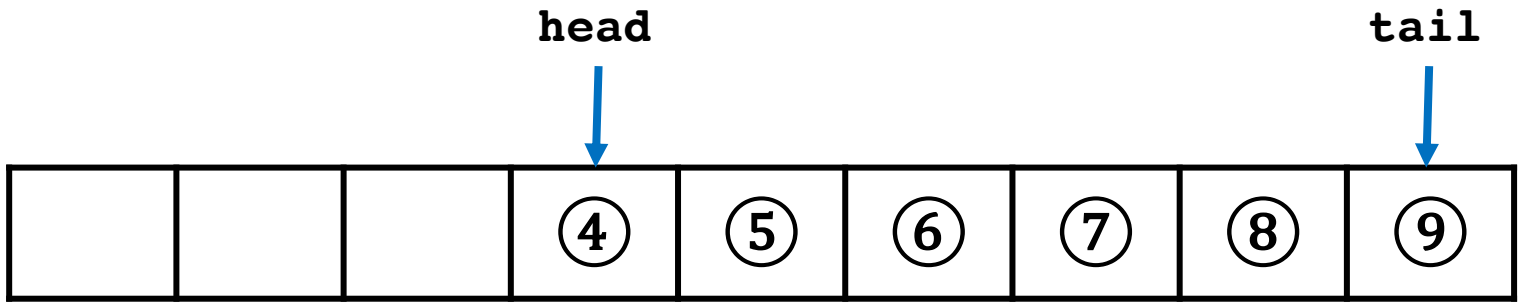
队列-假溢出



插入 ⑩ ？

1、length = 9
2、head = 3
3、tail = 8
4、data_type = xxx

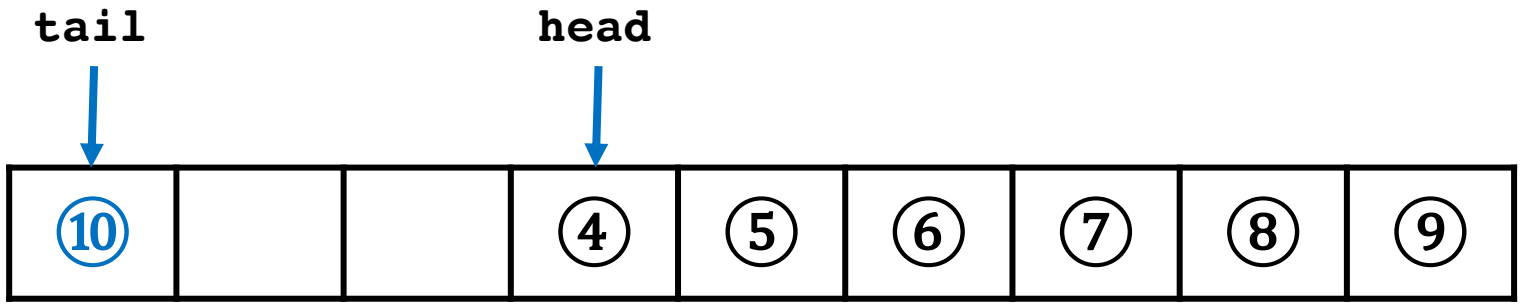
循环队列



插入 ⑩ ？

1、length = 9
2、head = 3
3、tail = 8
4、count = 6
5、data_type = xxx

循环队列



插入 ⑩ ？

1、length = 9
2、head = 3
3、tail = 8
4、count = 6
5、data_type = xxx

1、length = 9
2、head = 3
3、tail = 0
4、count = 7
5、data_type = xxx

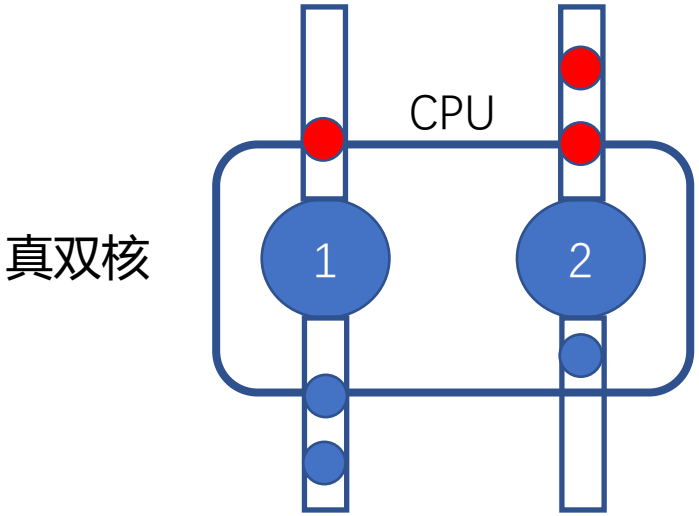
几种经典的队列实现方法

队列的典型应用场景

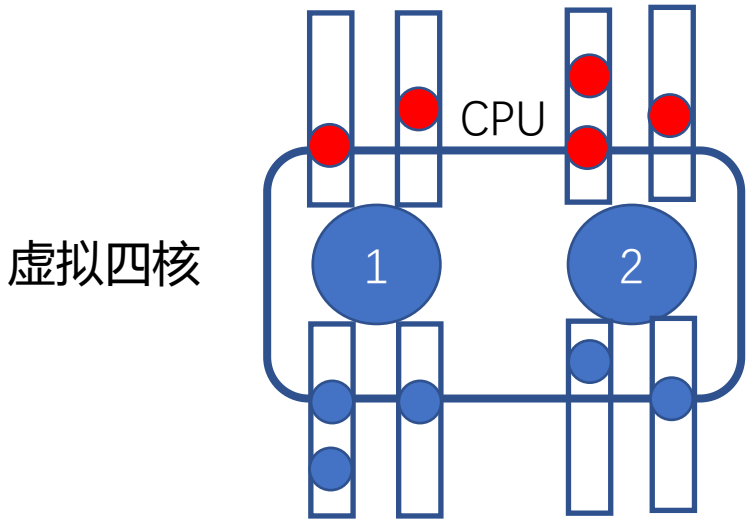
大约用时：（ 20 mins ）

下一部分：经典面试题-链表复习题

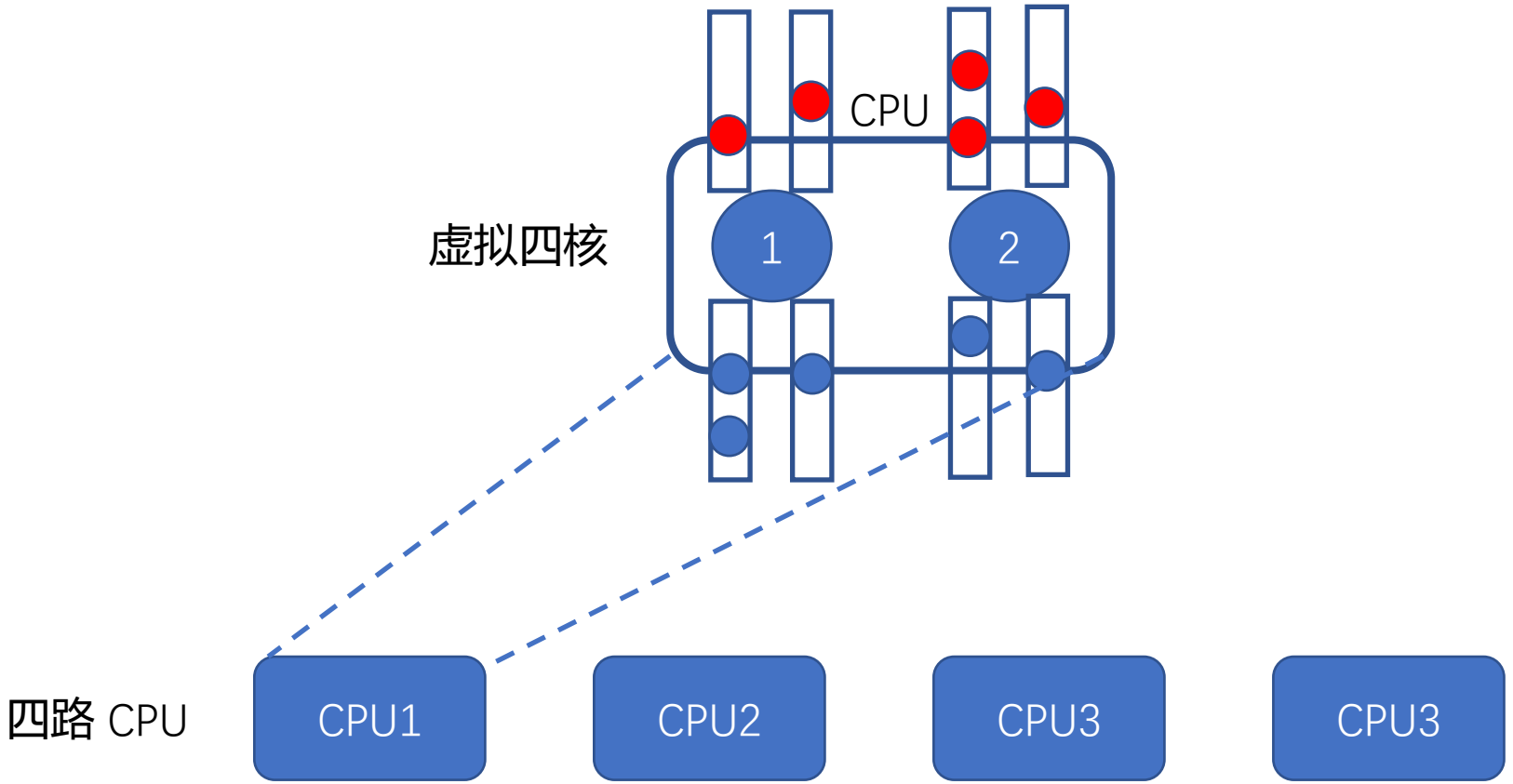
场景一：CPU 的超线程技术



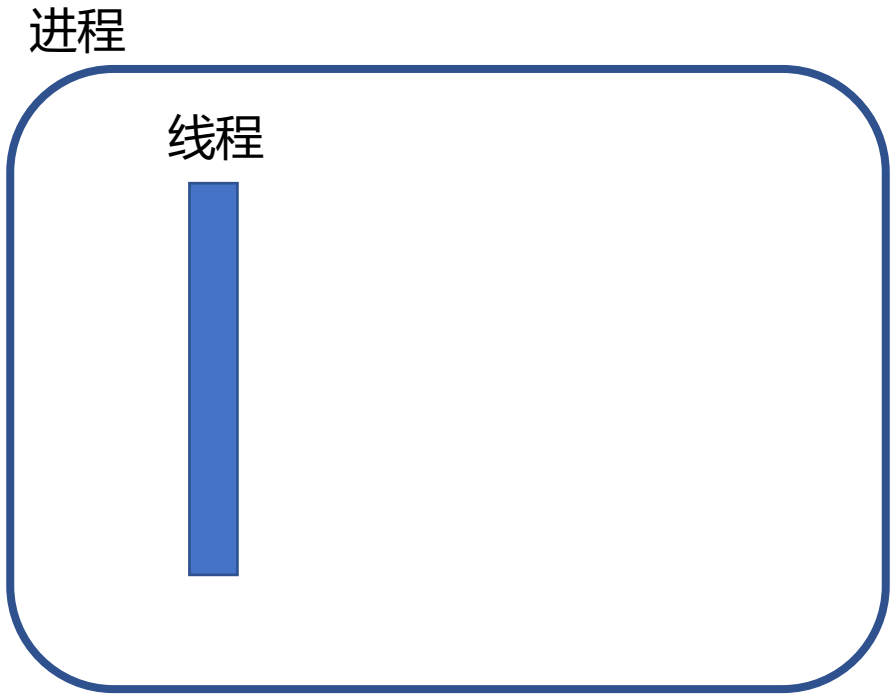
场景一：CPU 的超线程技术



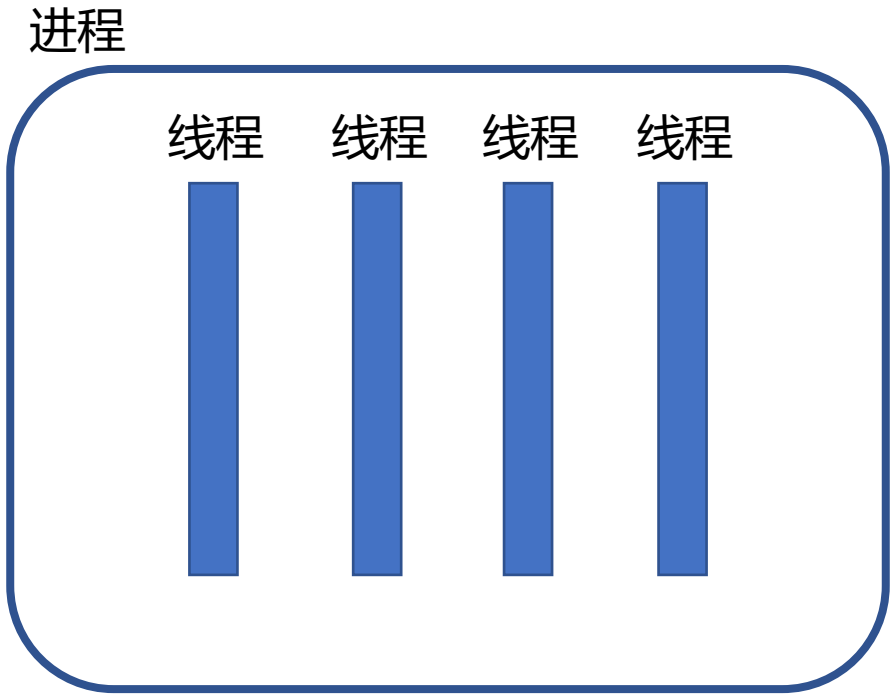
场景一：CPU 的超线程技术



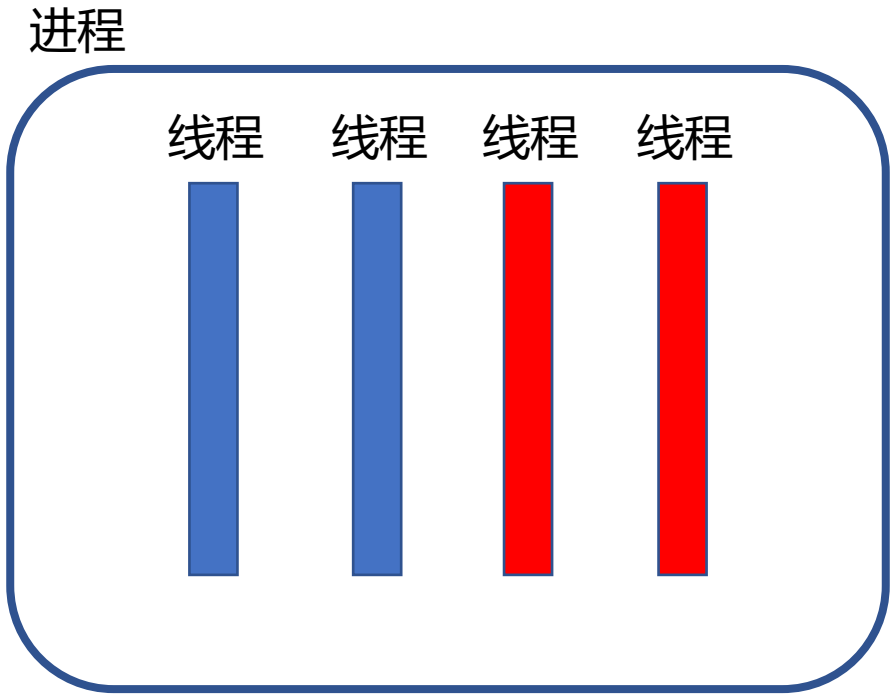
场景二：线程池的任务队列



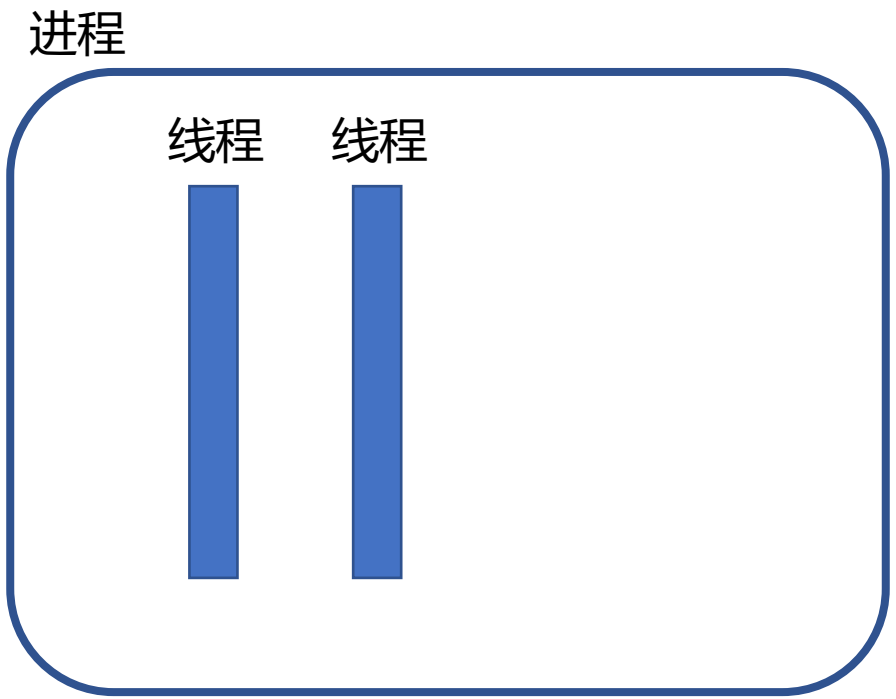
场景二：线程池的任务队列



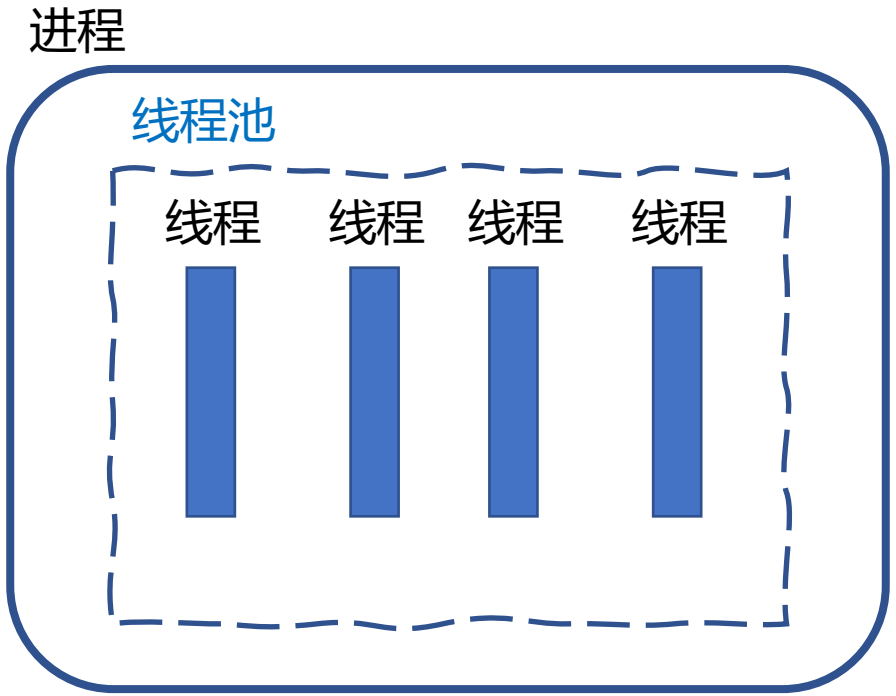
场景二：线程池的任务队列



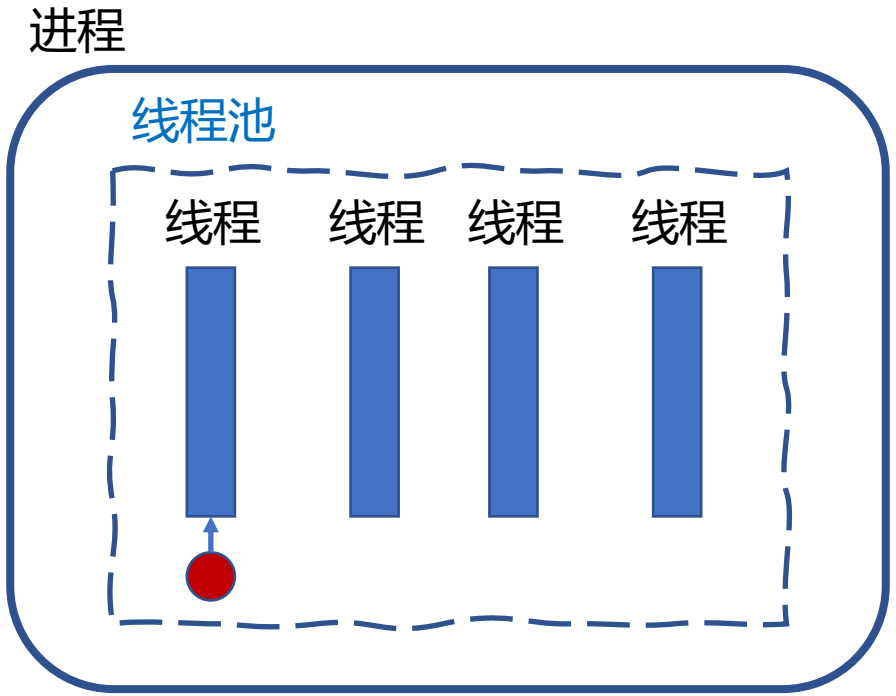
场景二：线程池的任务队列



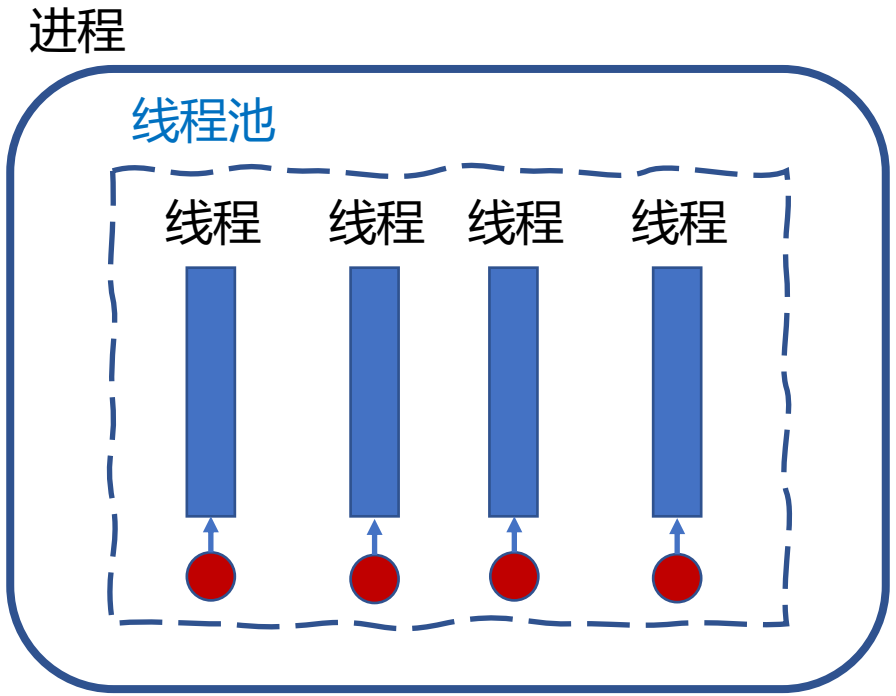
场景二：线程池的任务队列



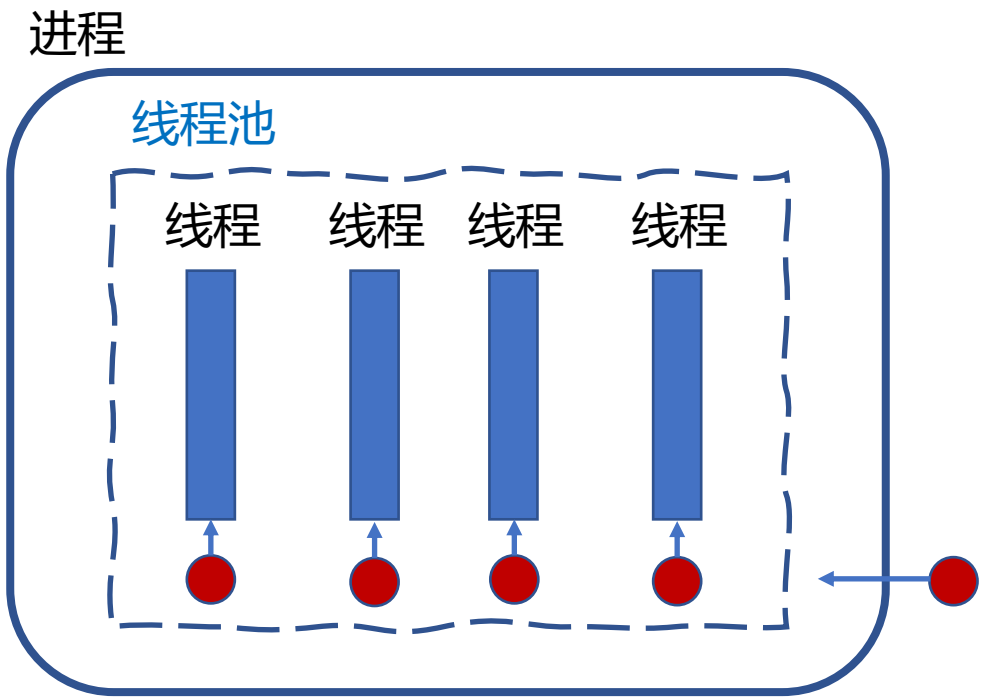
场景二：线程池的任务队列



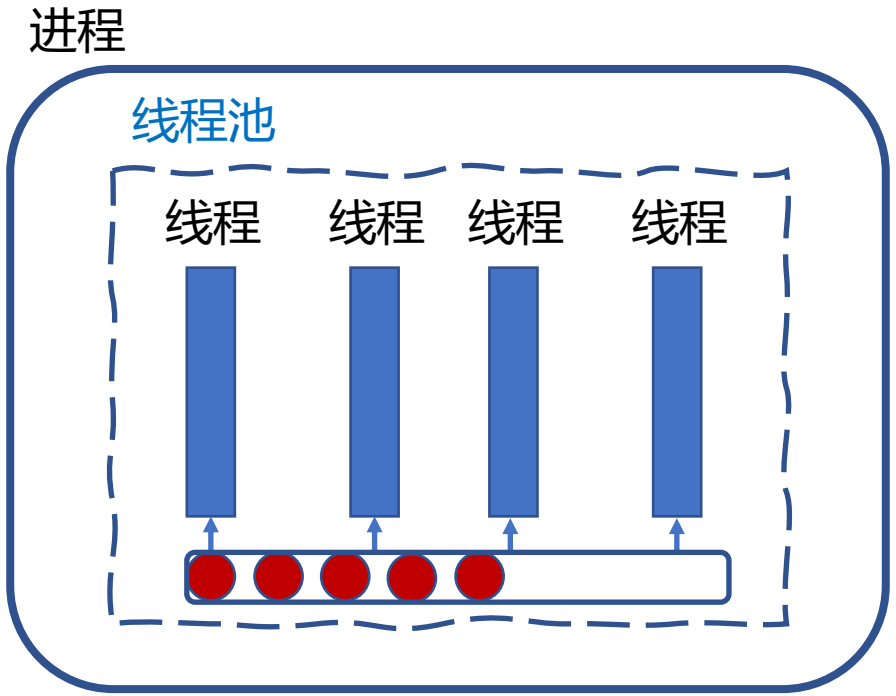
场景二：线程池的任务队列



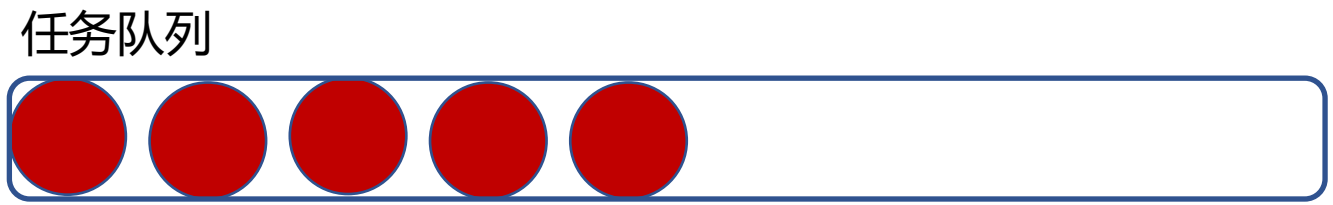
场景二：线程池的任务队列



场景二：线程池的任务队列



场景二：线程池的任务队列



经典面试题-链表复习题

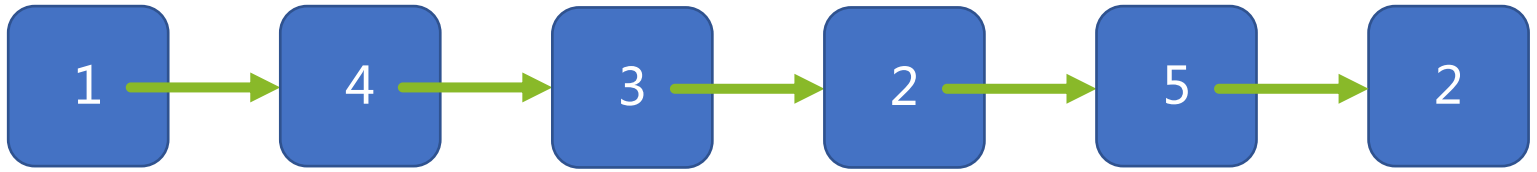
大约用时：（ 20 mins ）

下一部分：经典面试题-队列的封装与使用

86.分隔链表

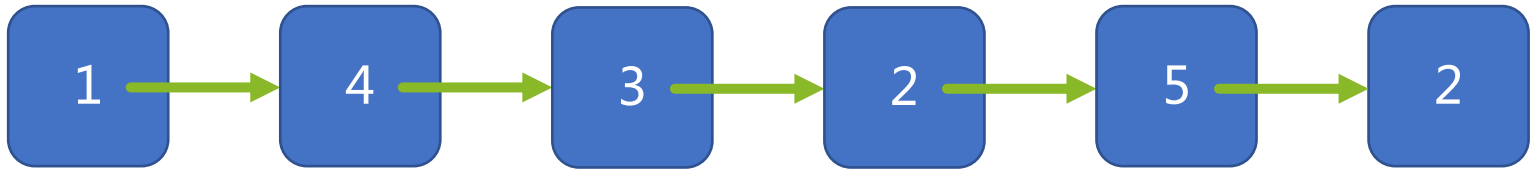
门徒计划，带你开启算法精进之路

LeetCode-86 分隔链表



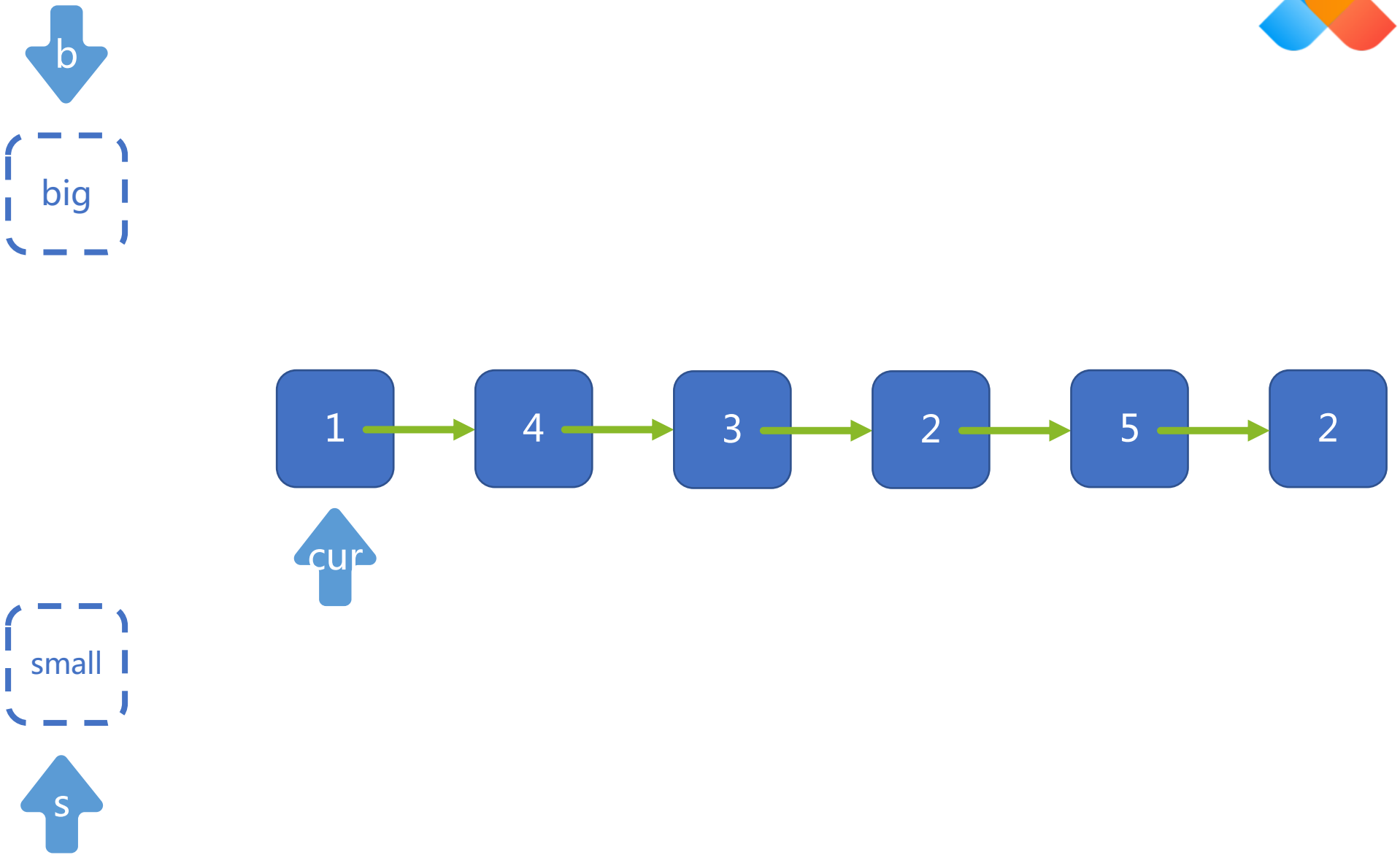
LeetCode-86 分隔链表

big

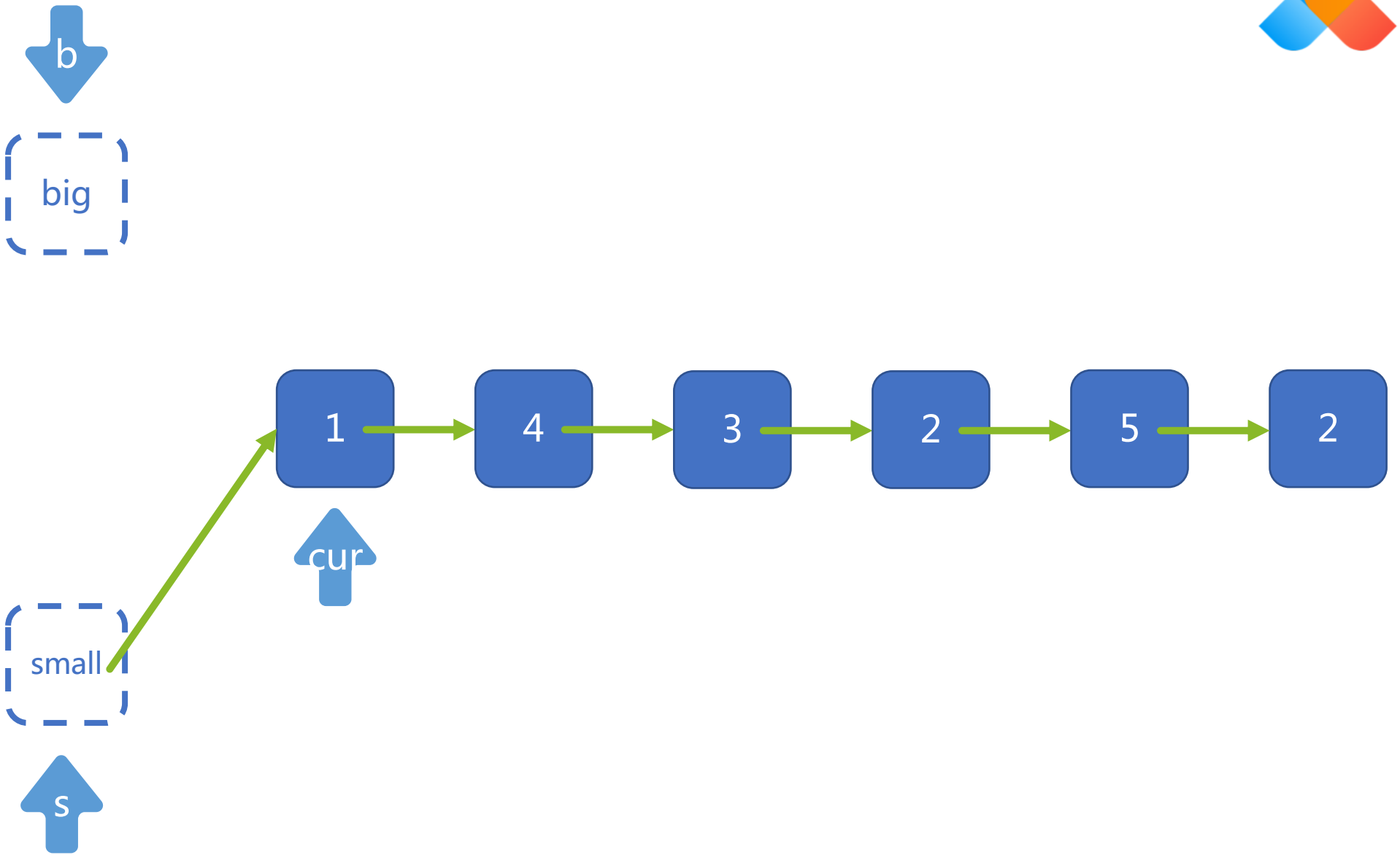


small

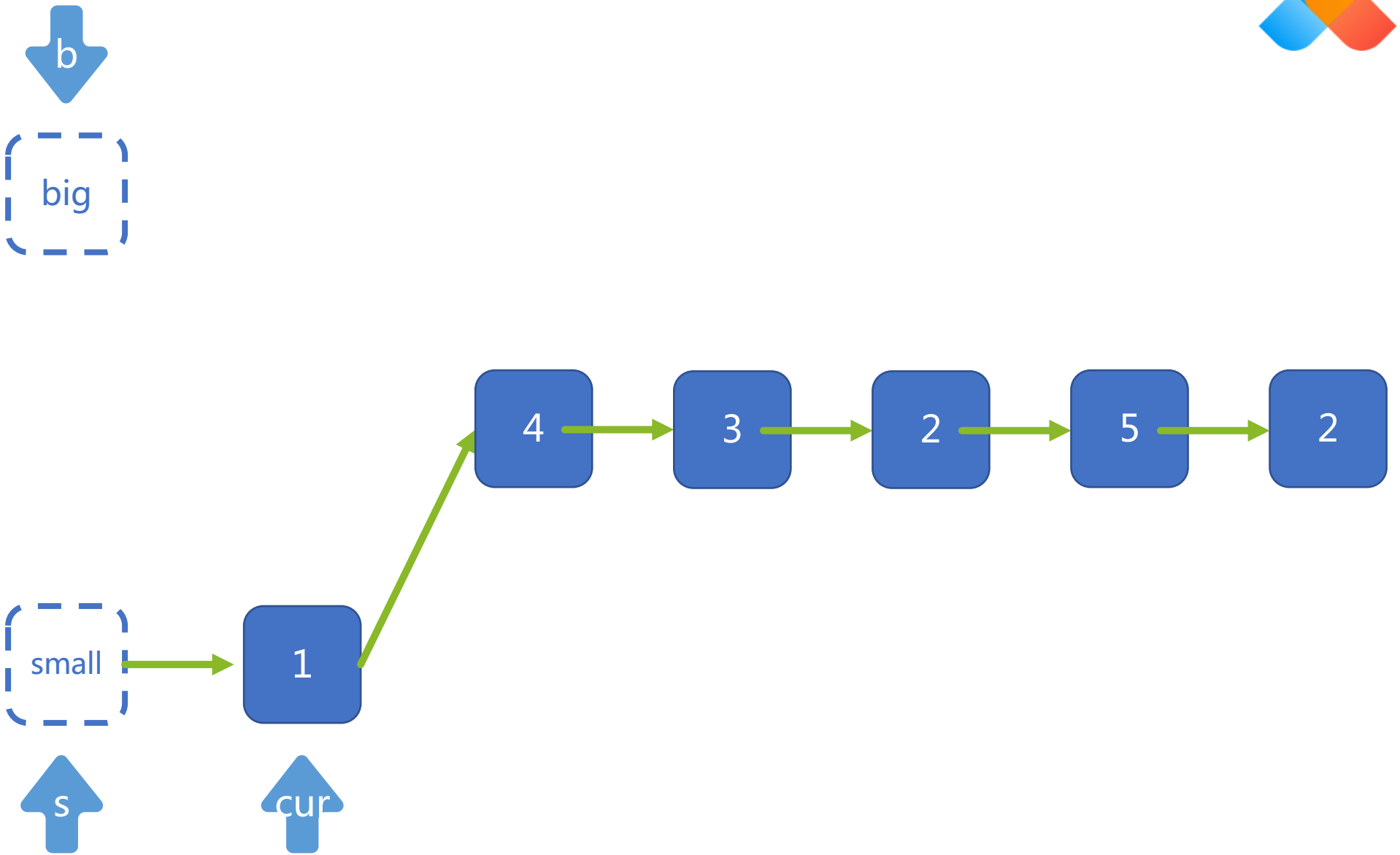
LeetCode-86 分隔链表



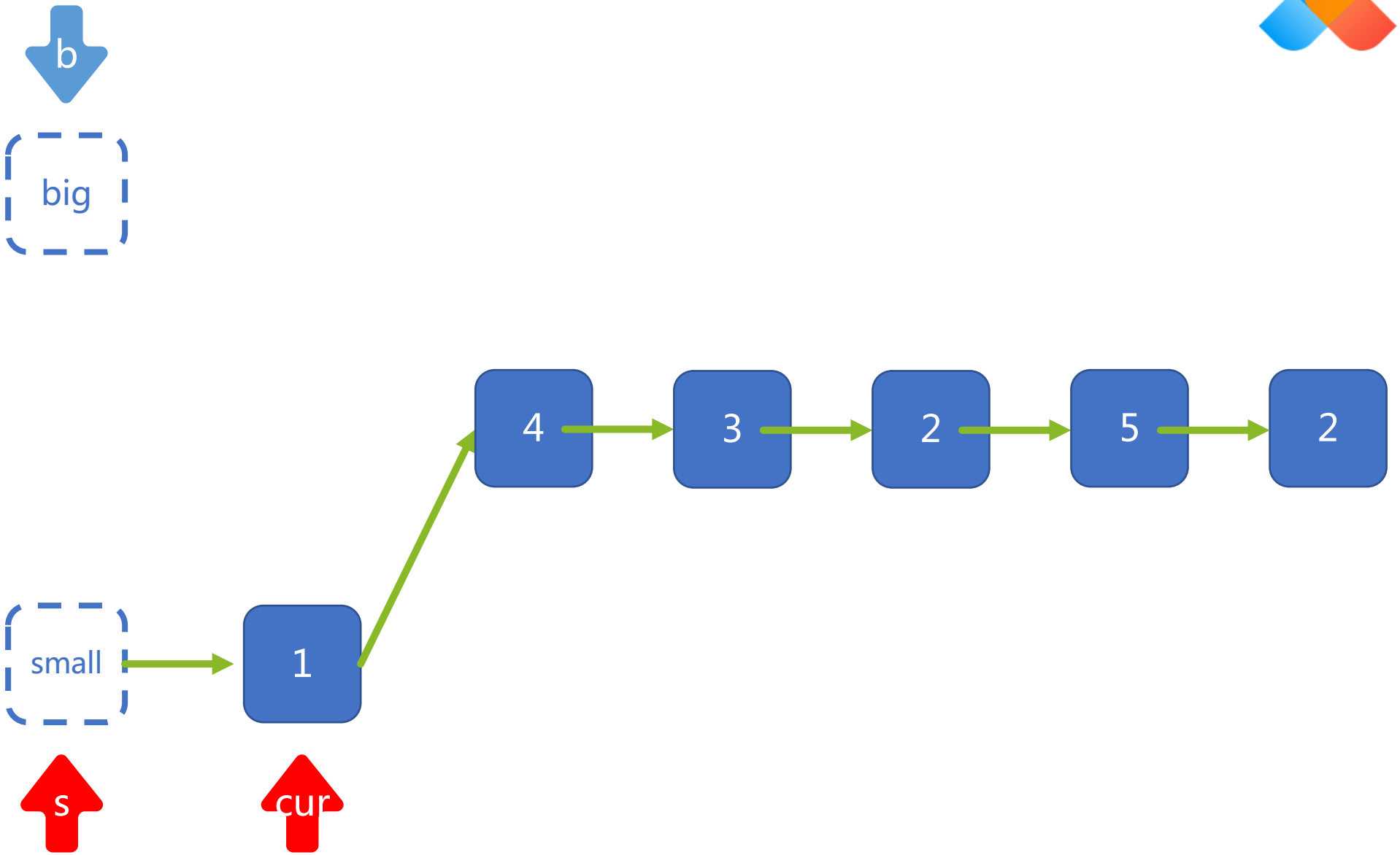
LeetCode-86 分隔链表



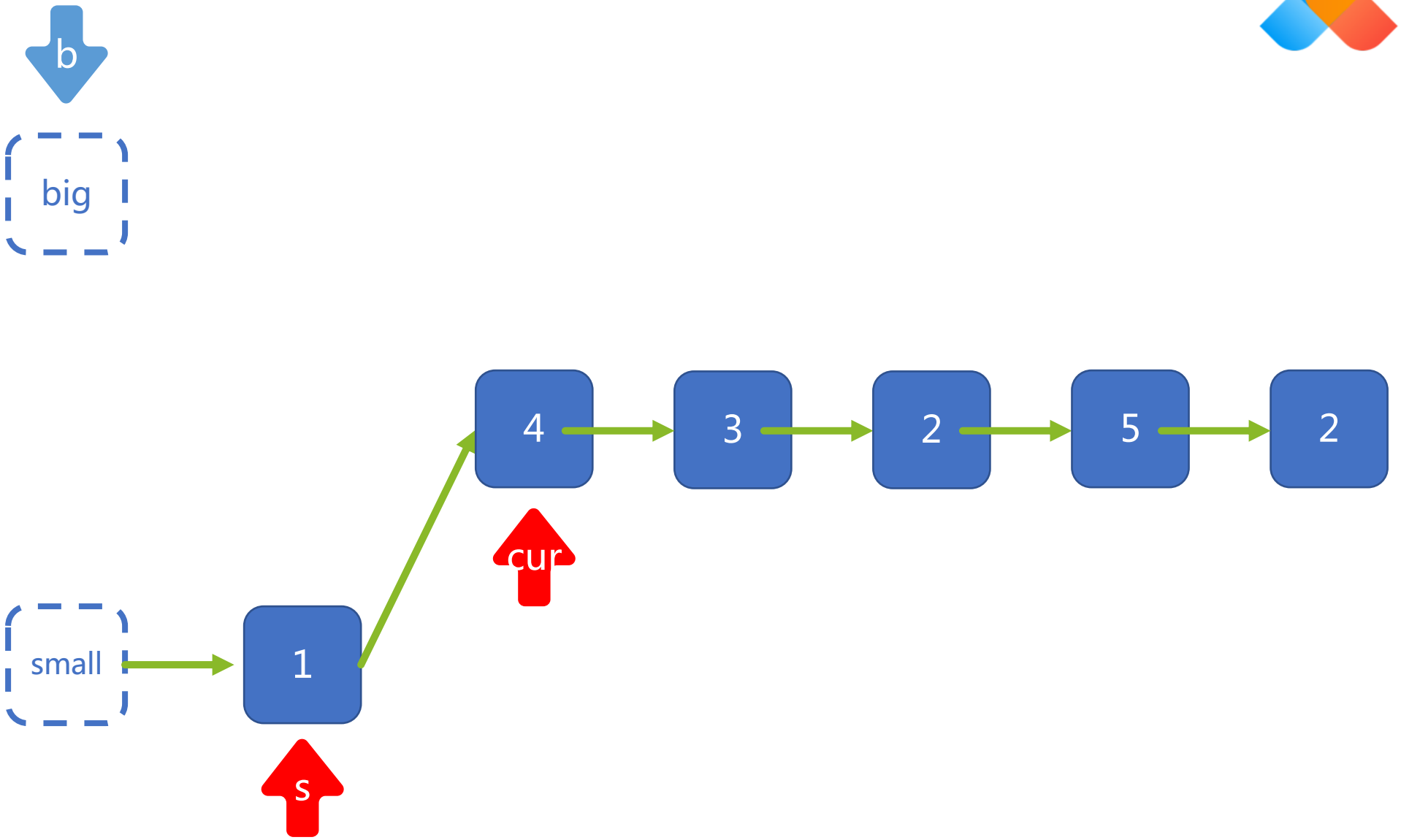
LeetCode-86 分隔链表



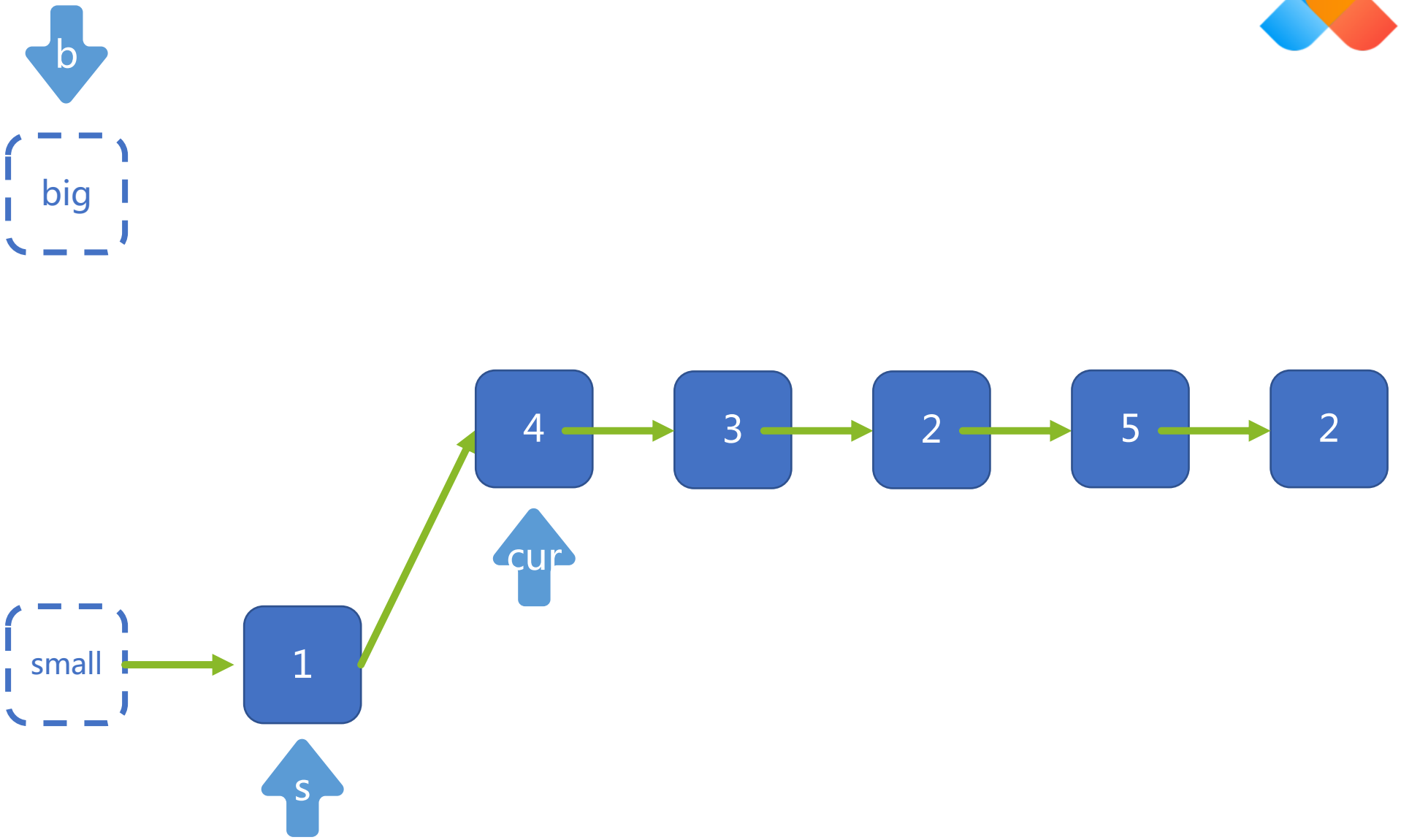
LeetCode-86 分隔链表



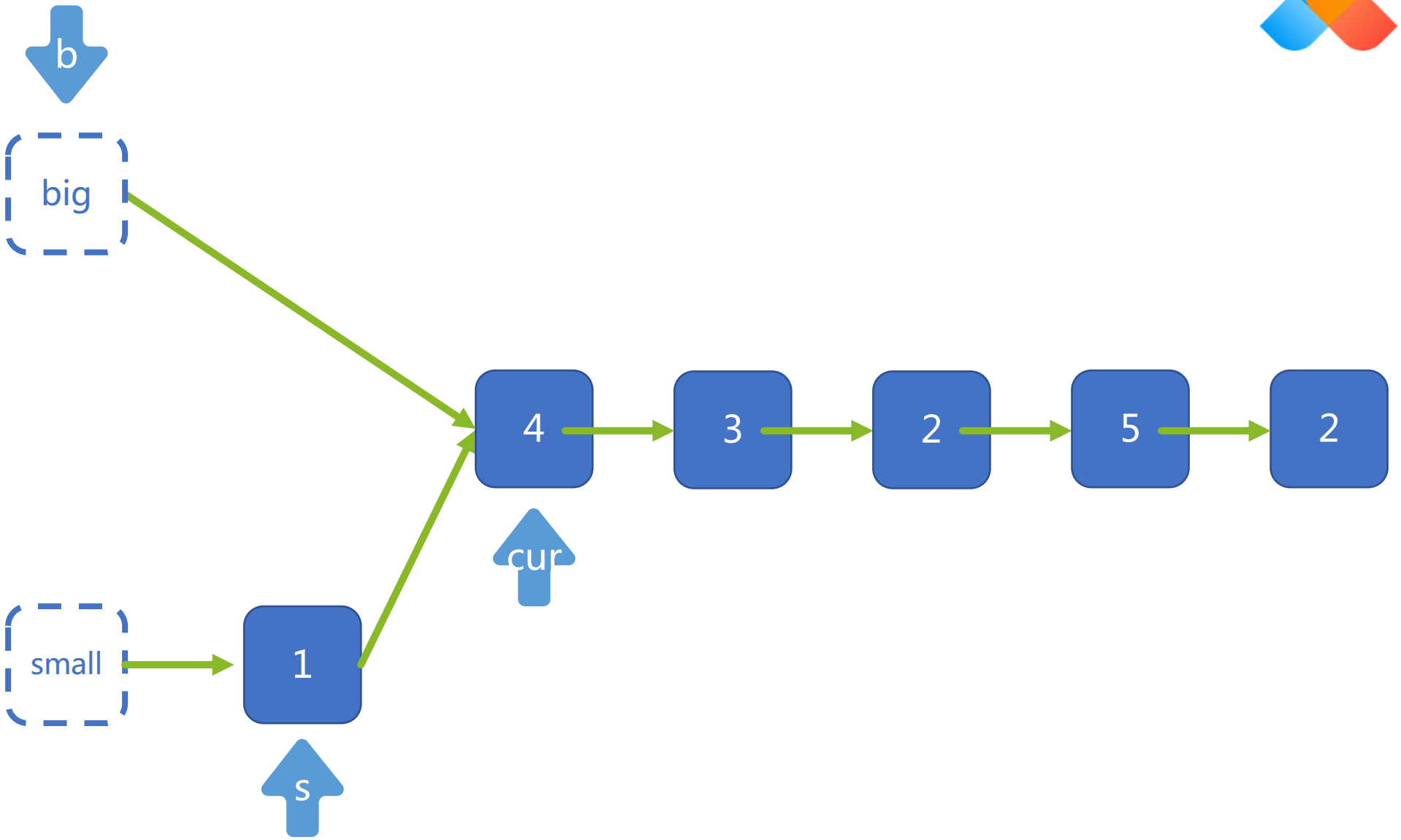
LeetCode-86 分隔链表



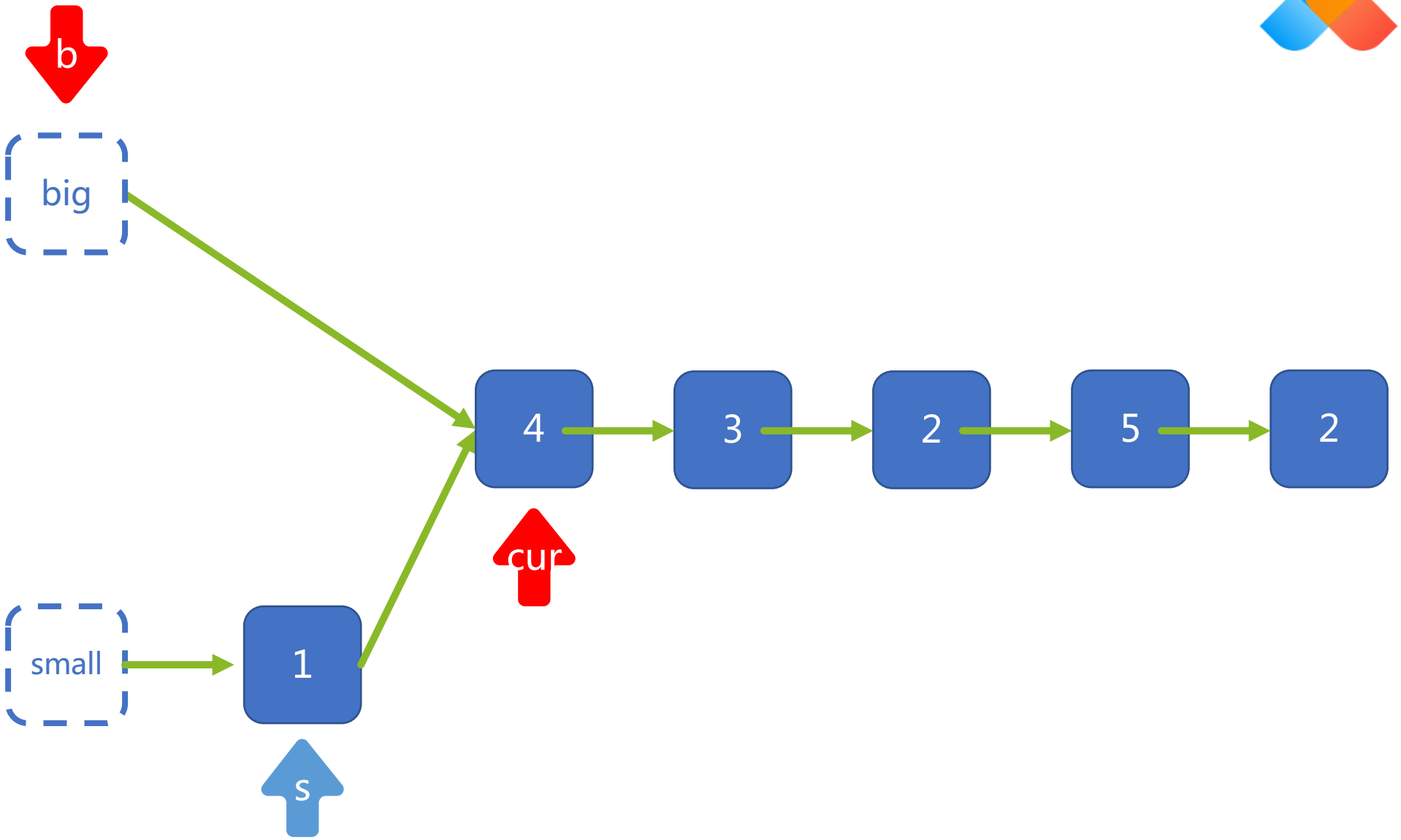
LeetCode-86 分隔链表



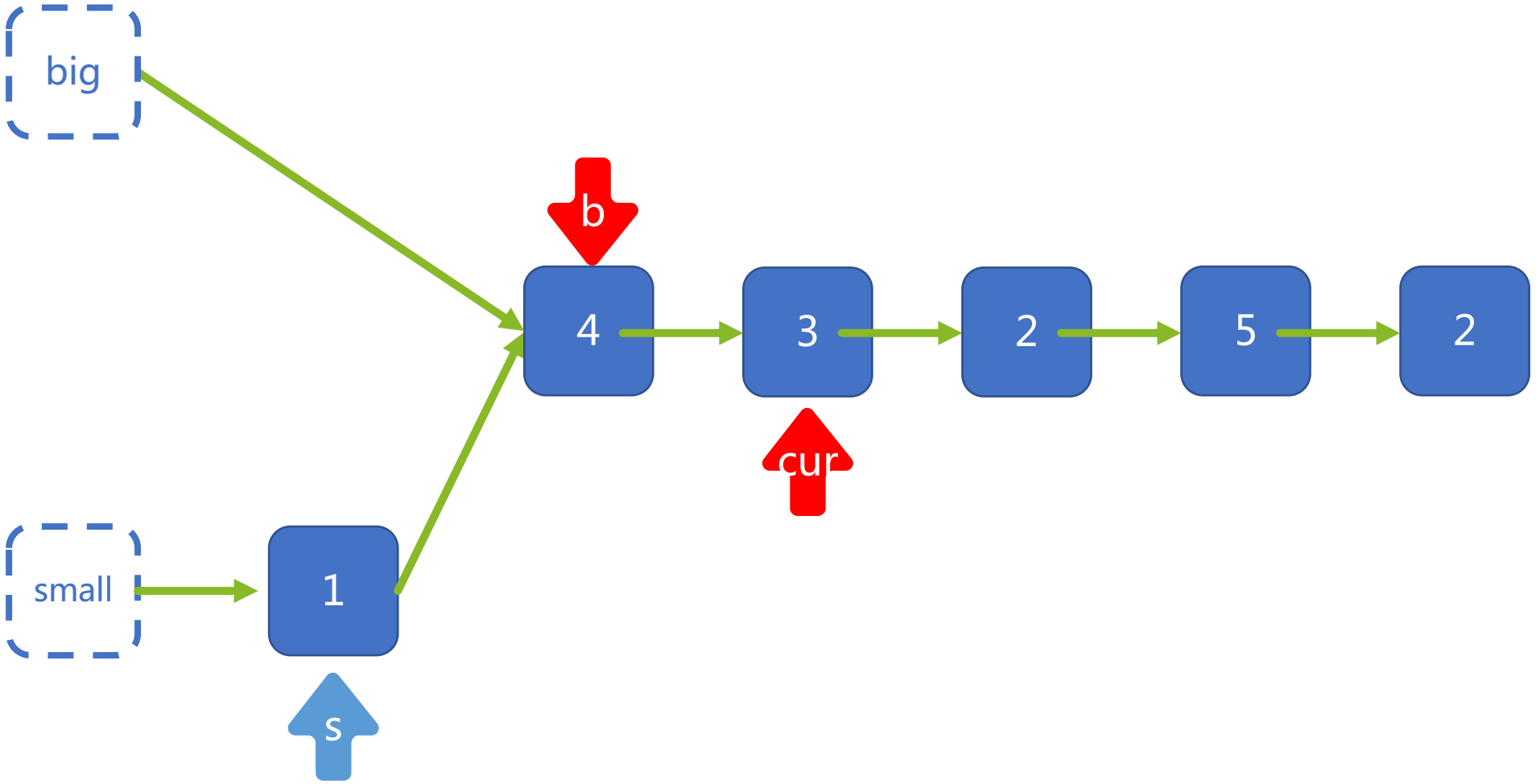
LeetCode-86 分隔链表



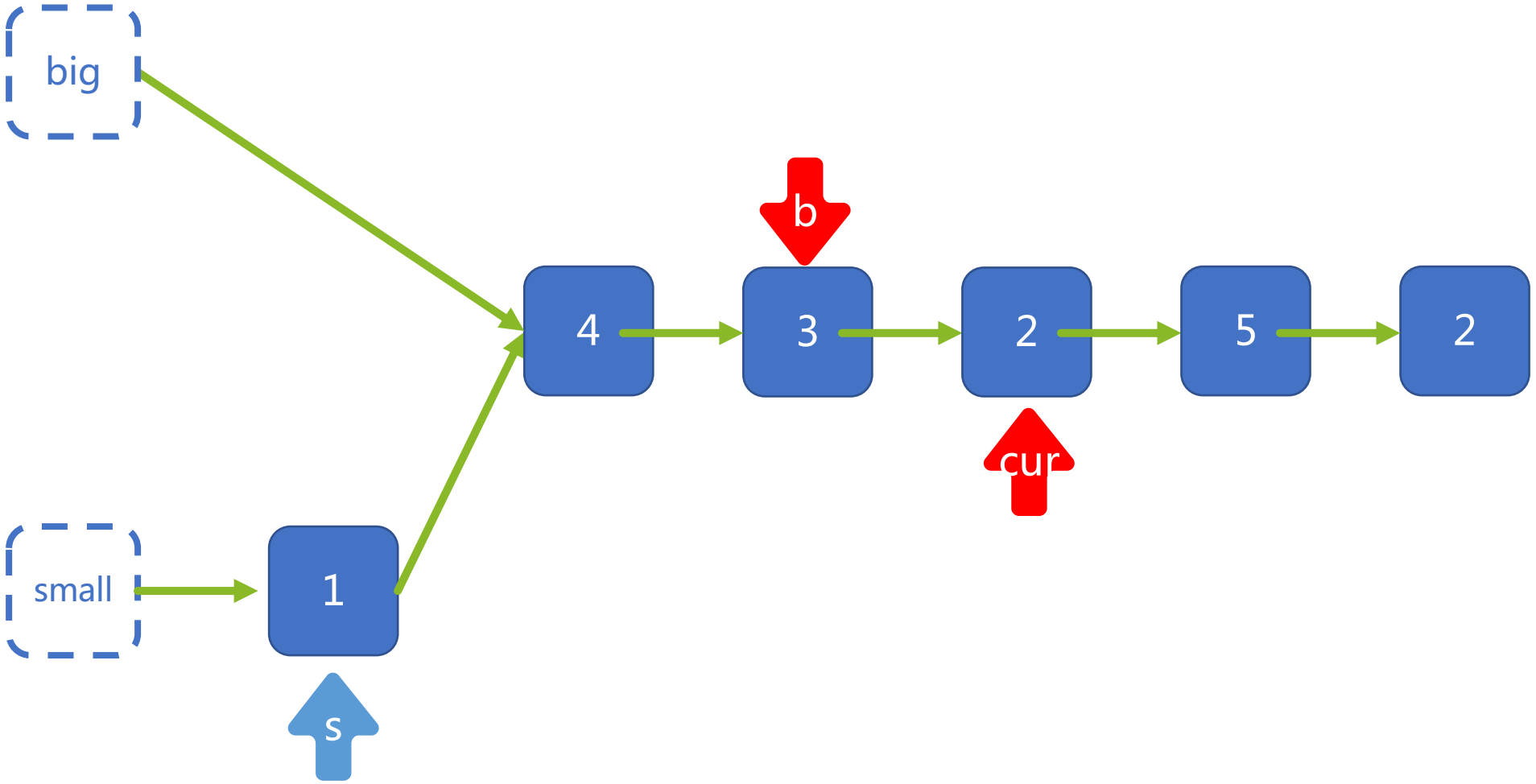
LeetCode-86 分隔链表



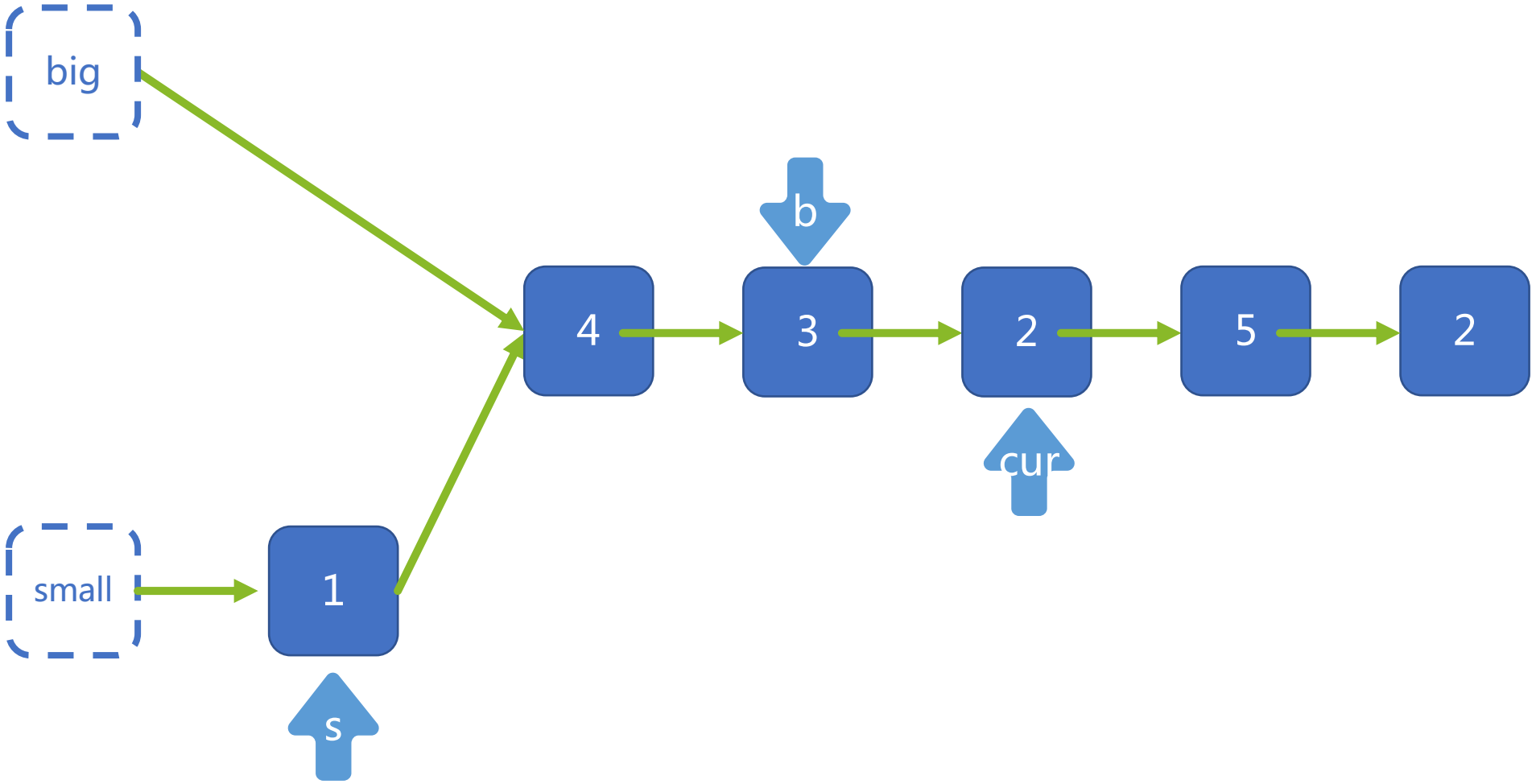
LeetCode-86 分隔链表



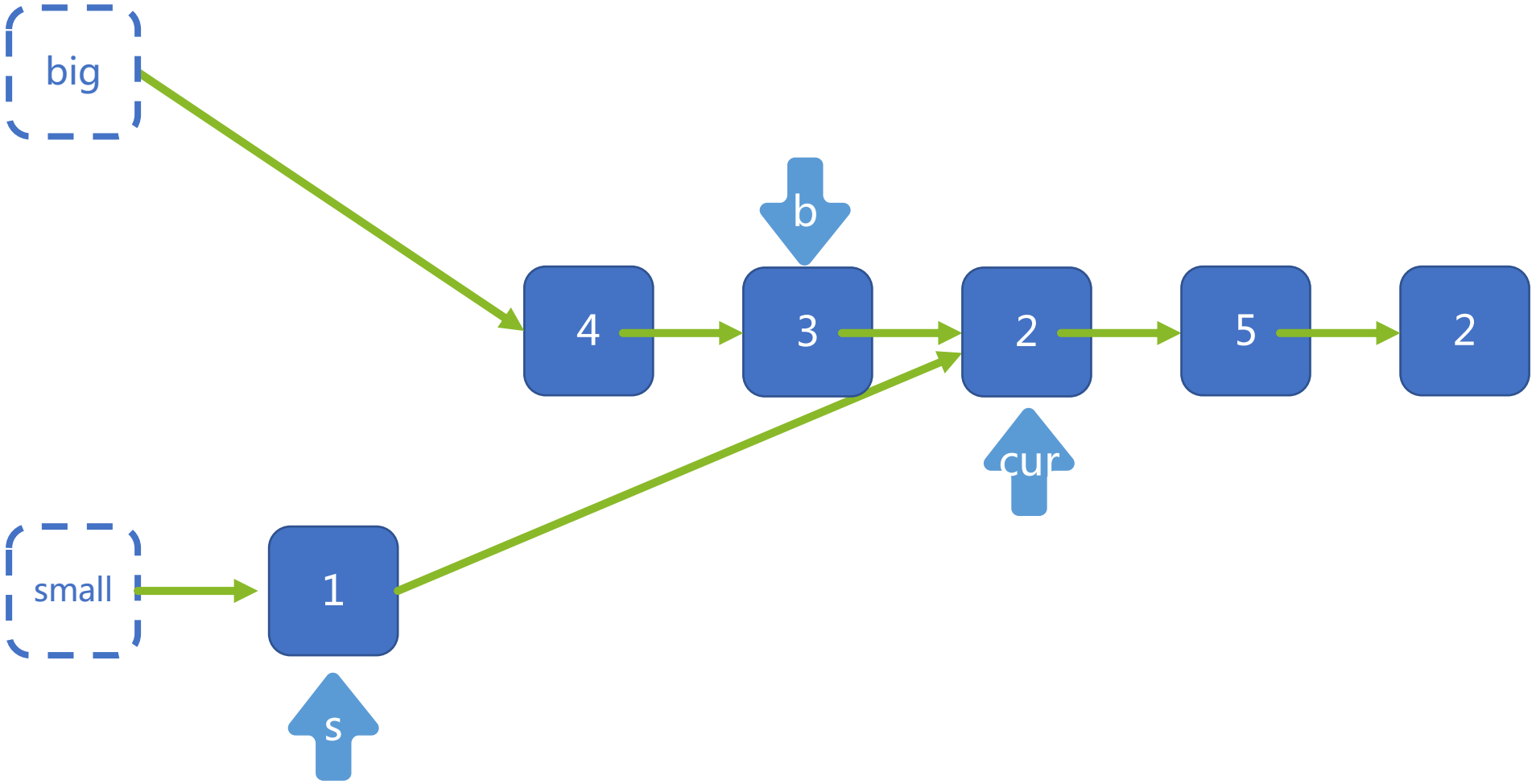
LeetCode-86 分隔链表



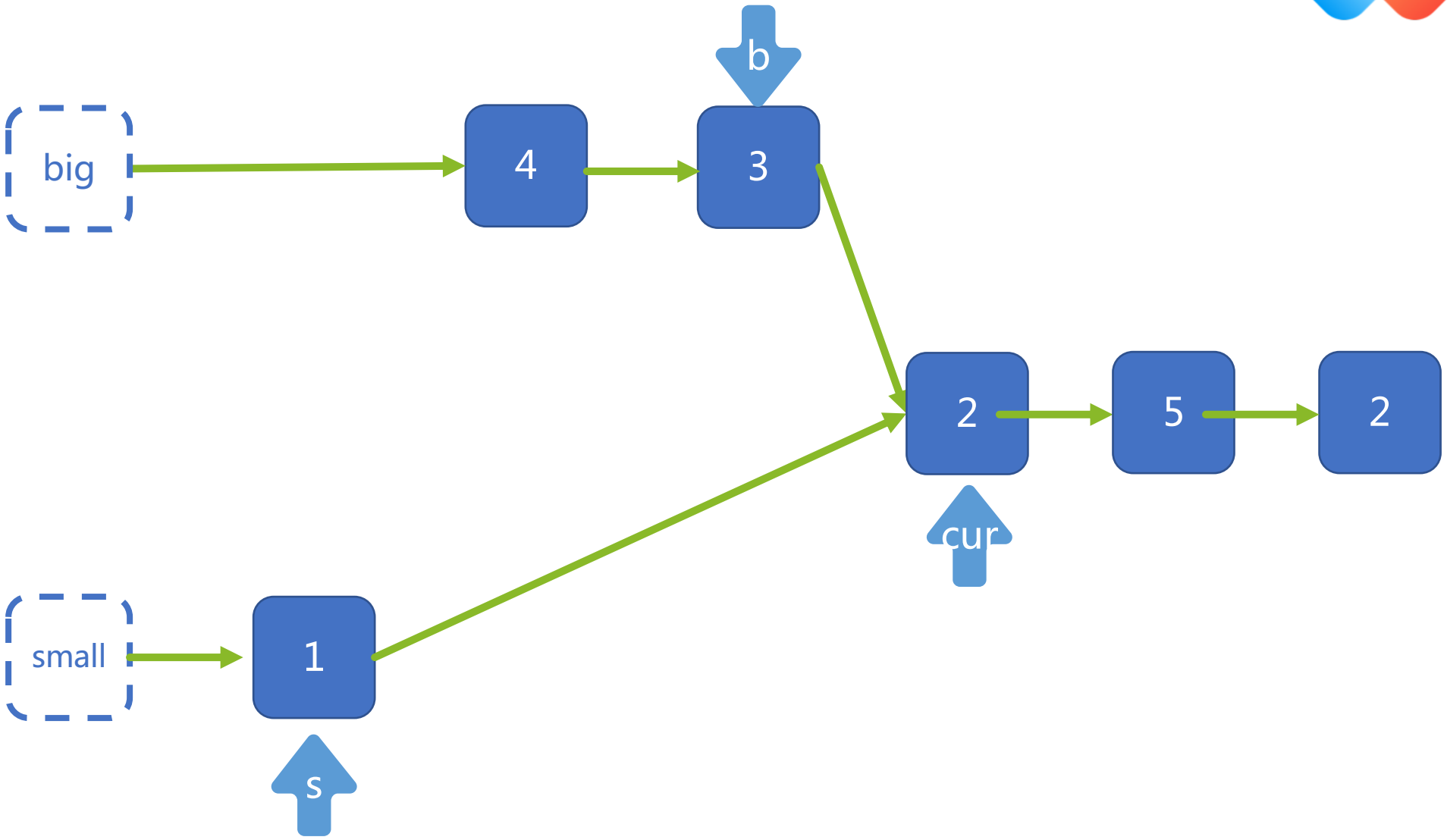
LeetCode-86 分隔链表



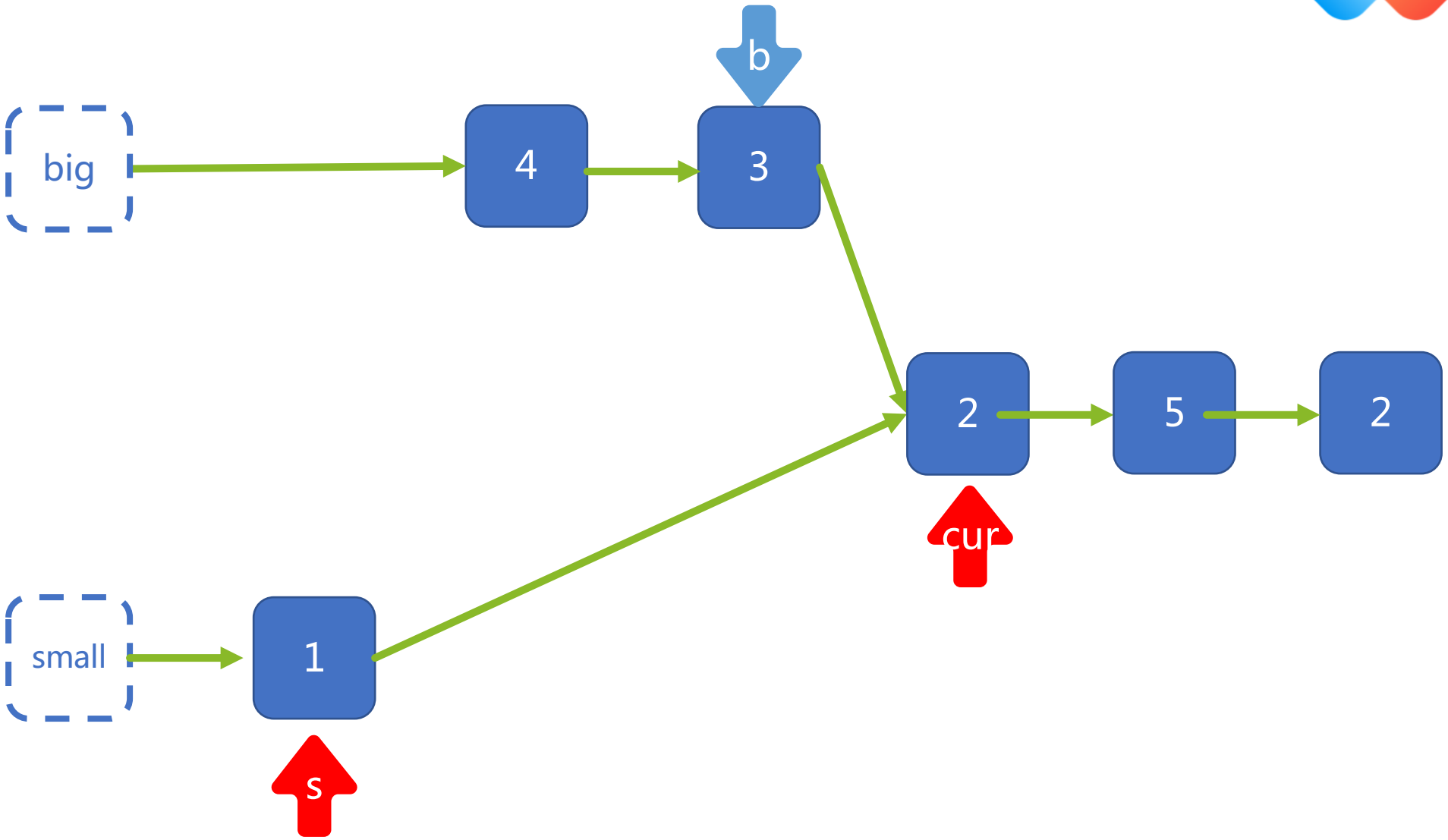
LeetCode-86 分隔链表



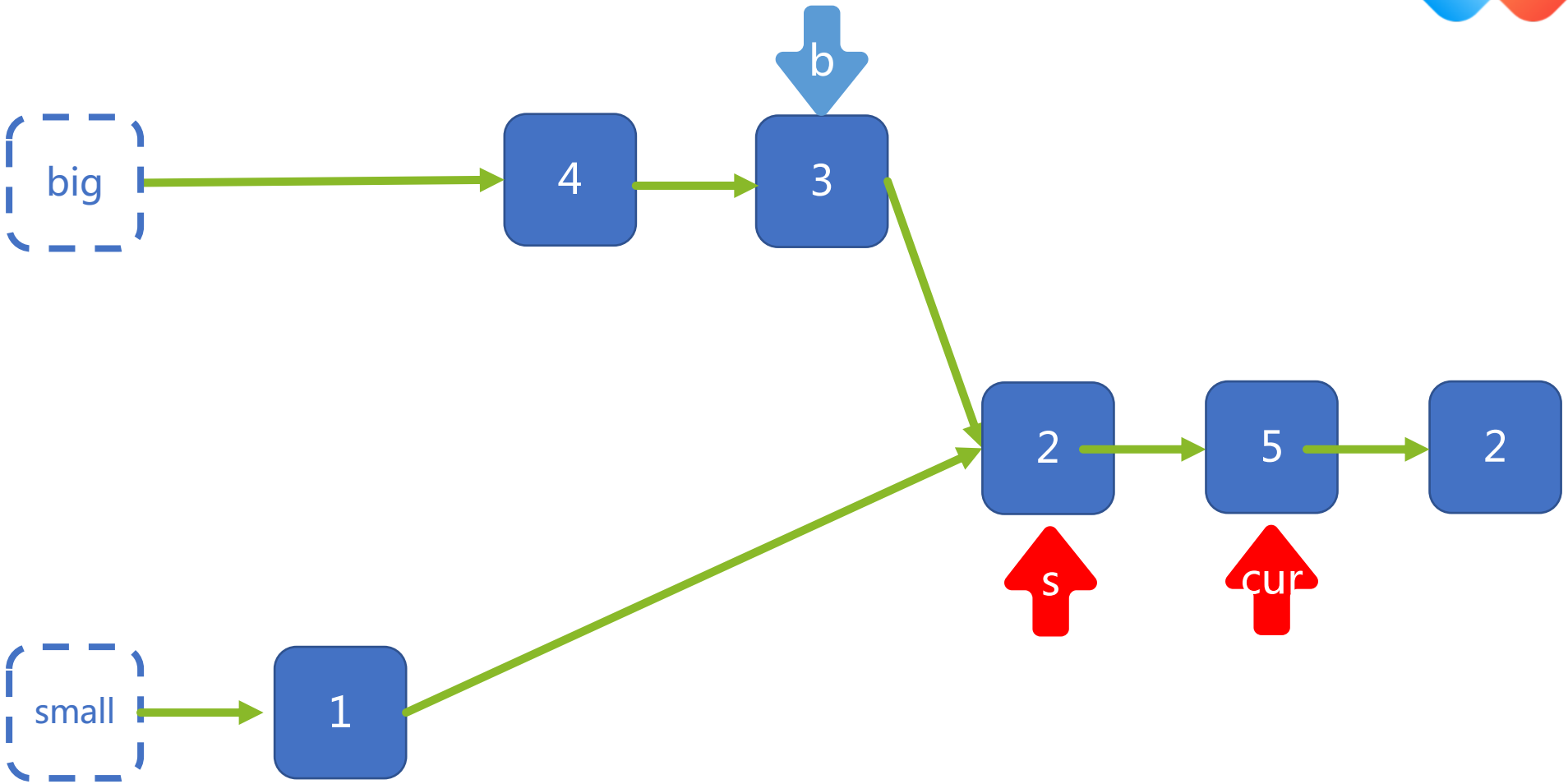
LeetCode-86 分隔链表



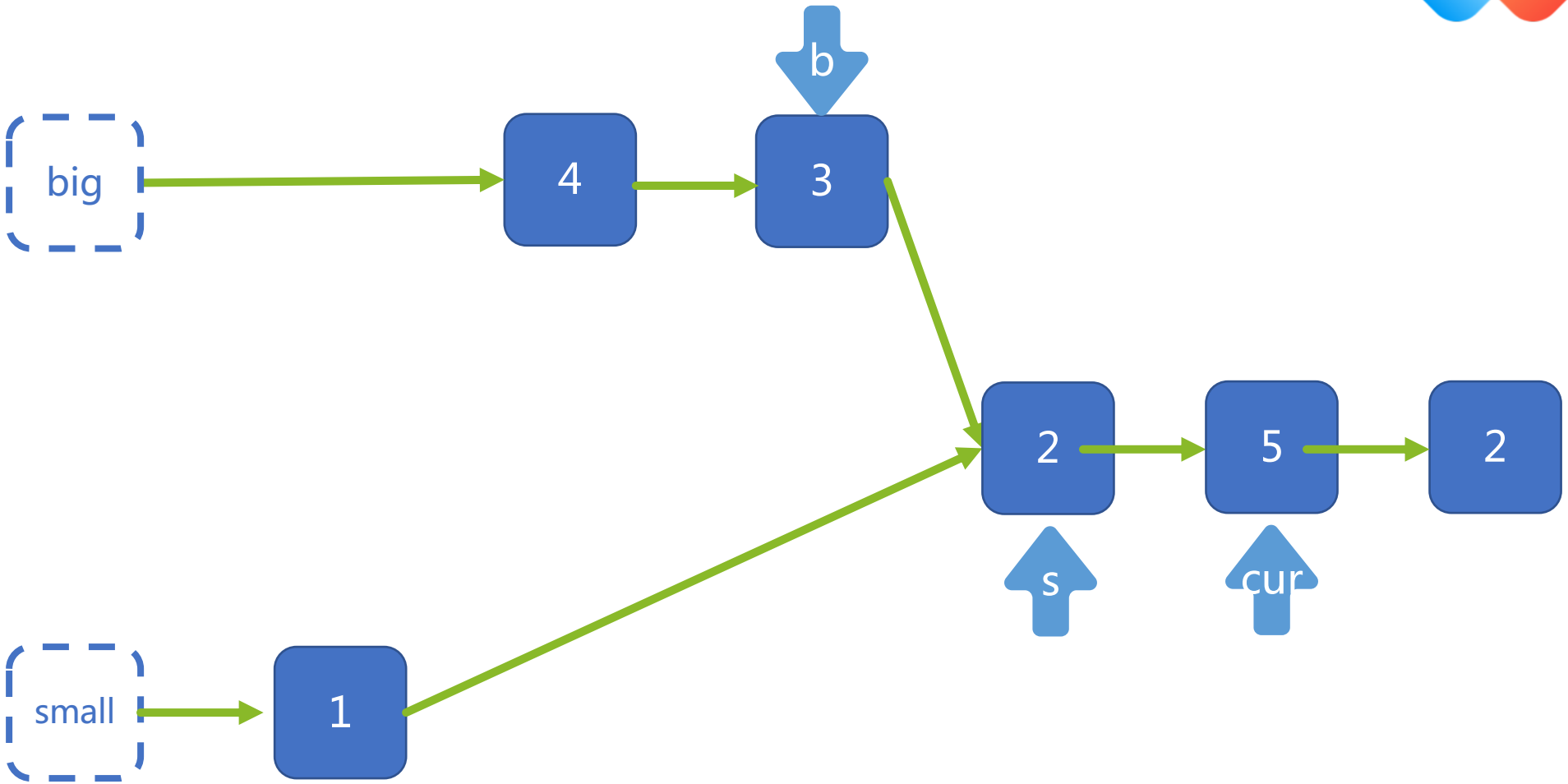
LeetCode-86 分隔链表



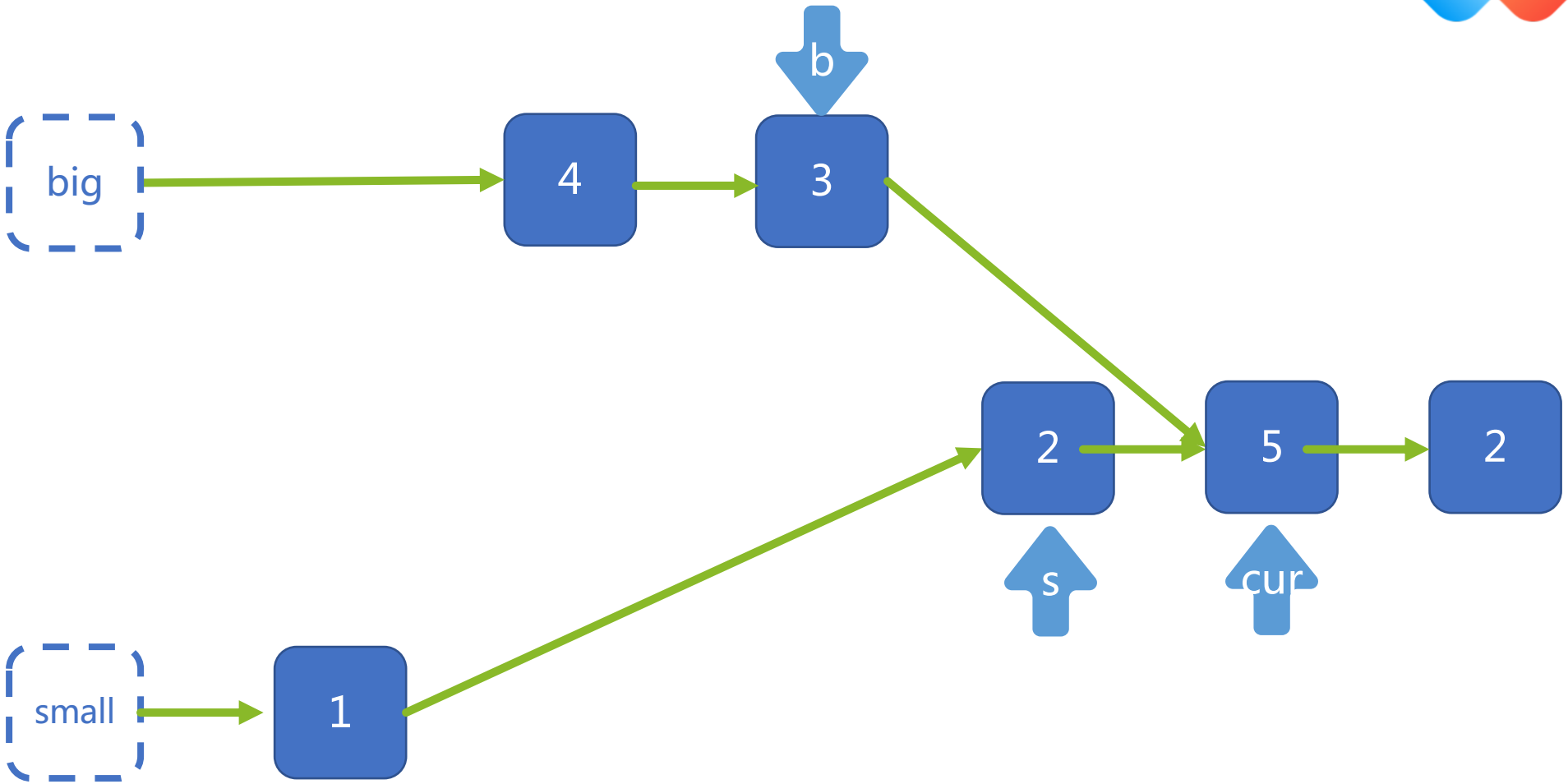
LeetCode-86 分隔链表



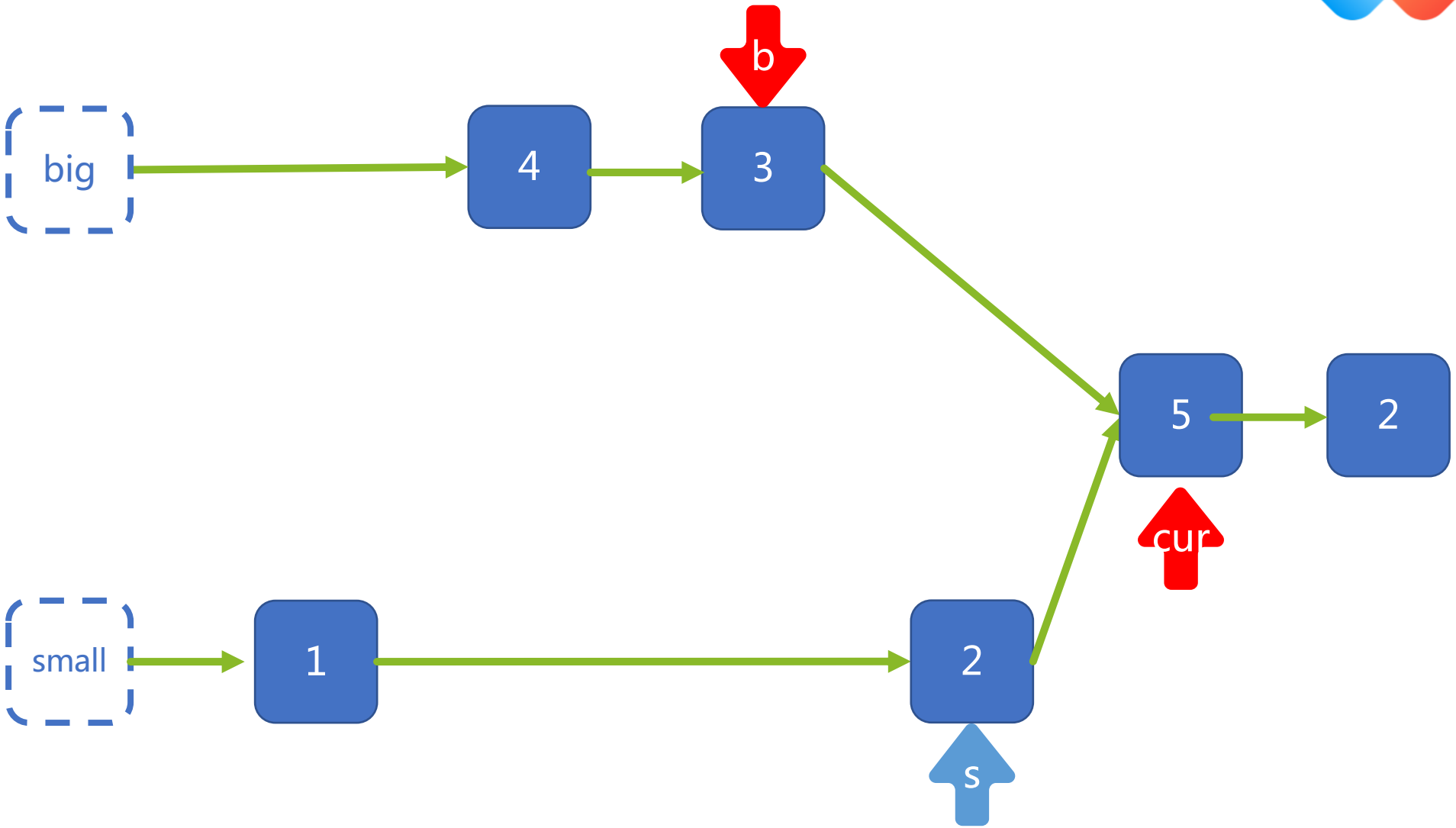
LeetCode-86 分隔链表



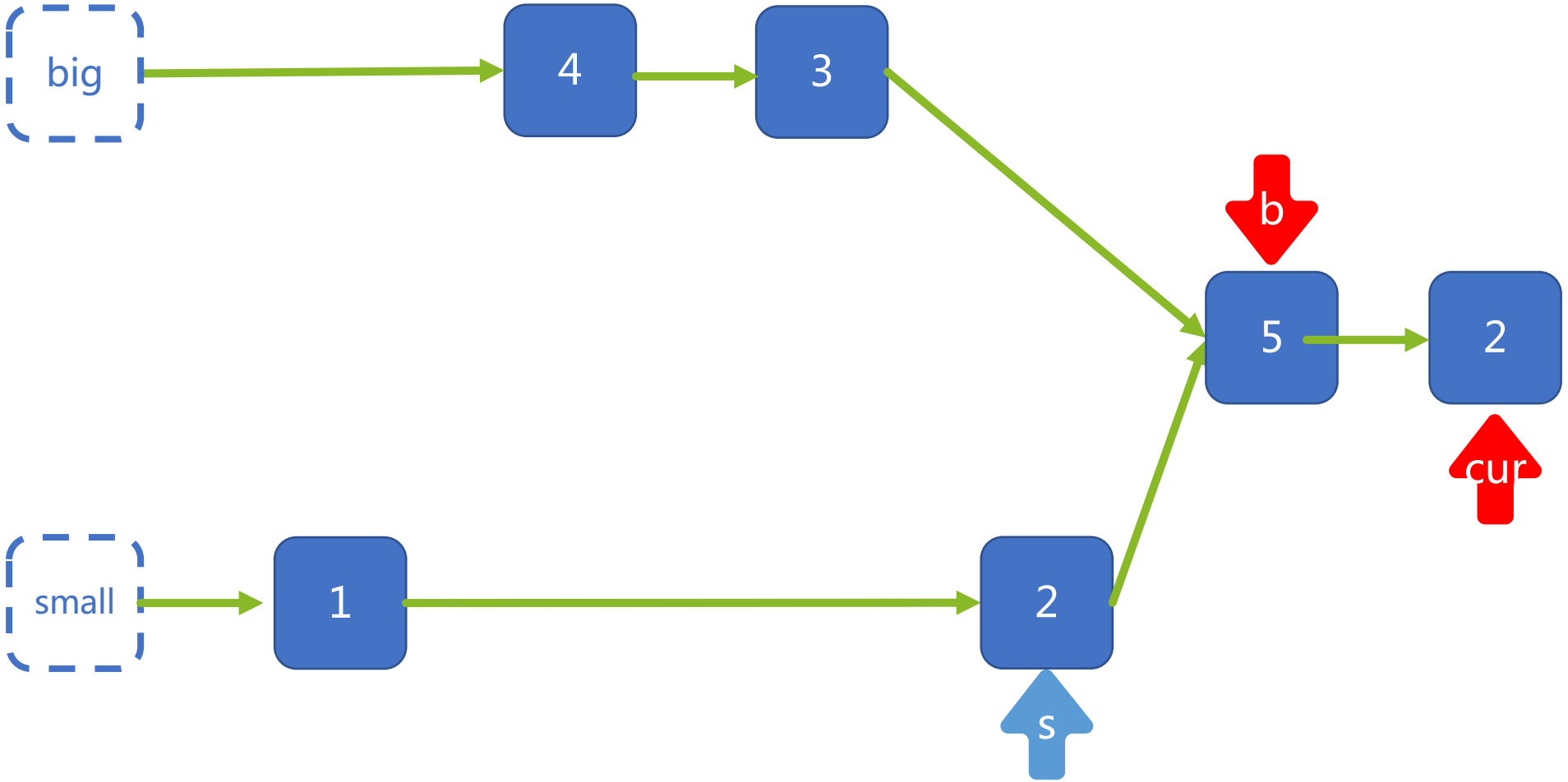
LeetCode-86 分隔链表



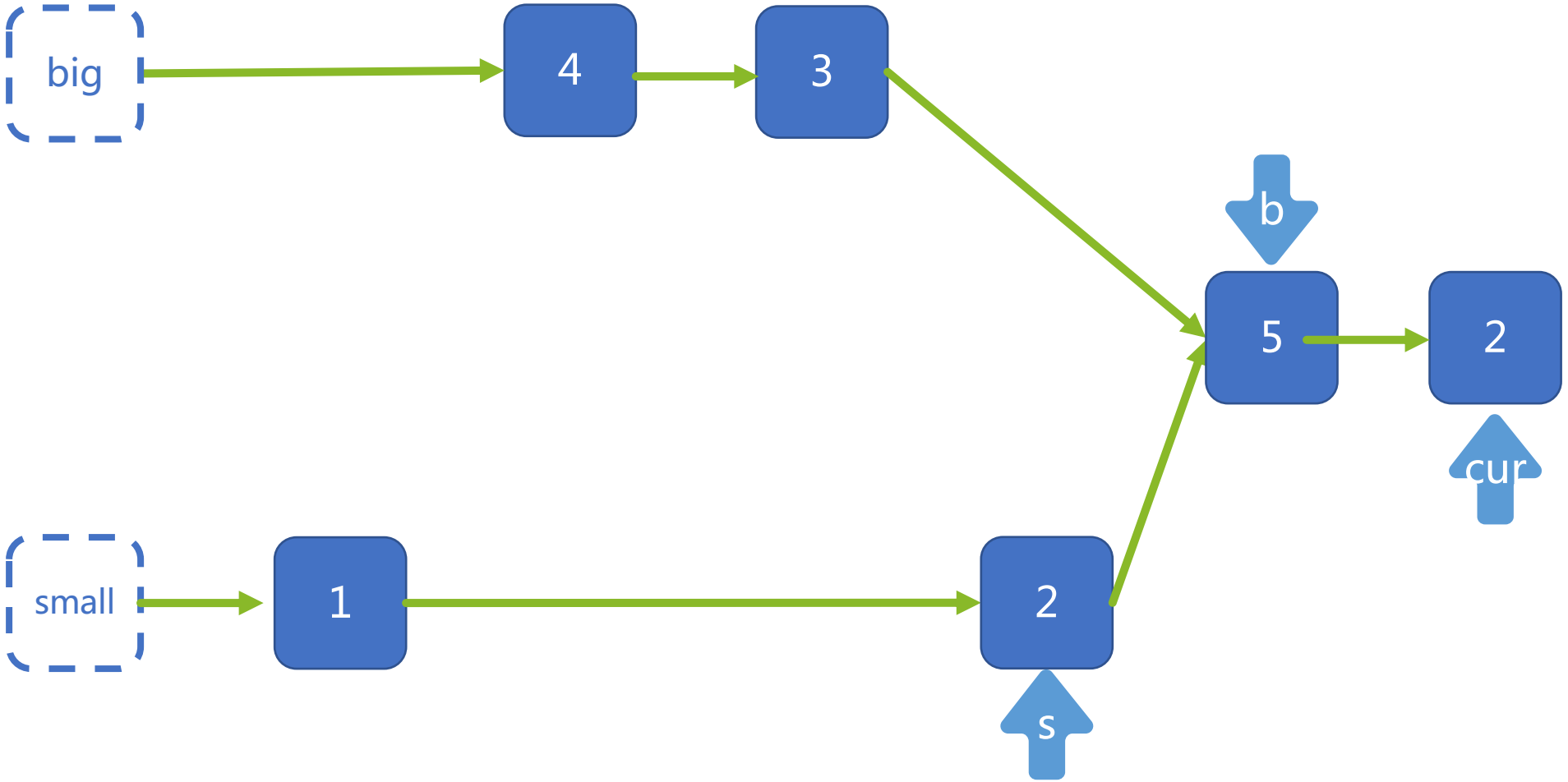
LeetCode-86 分隔链表



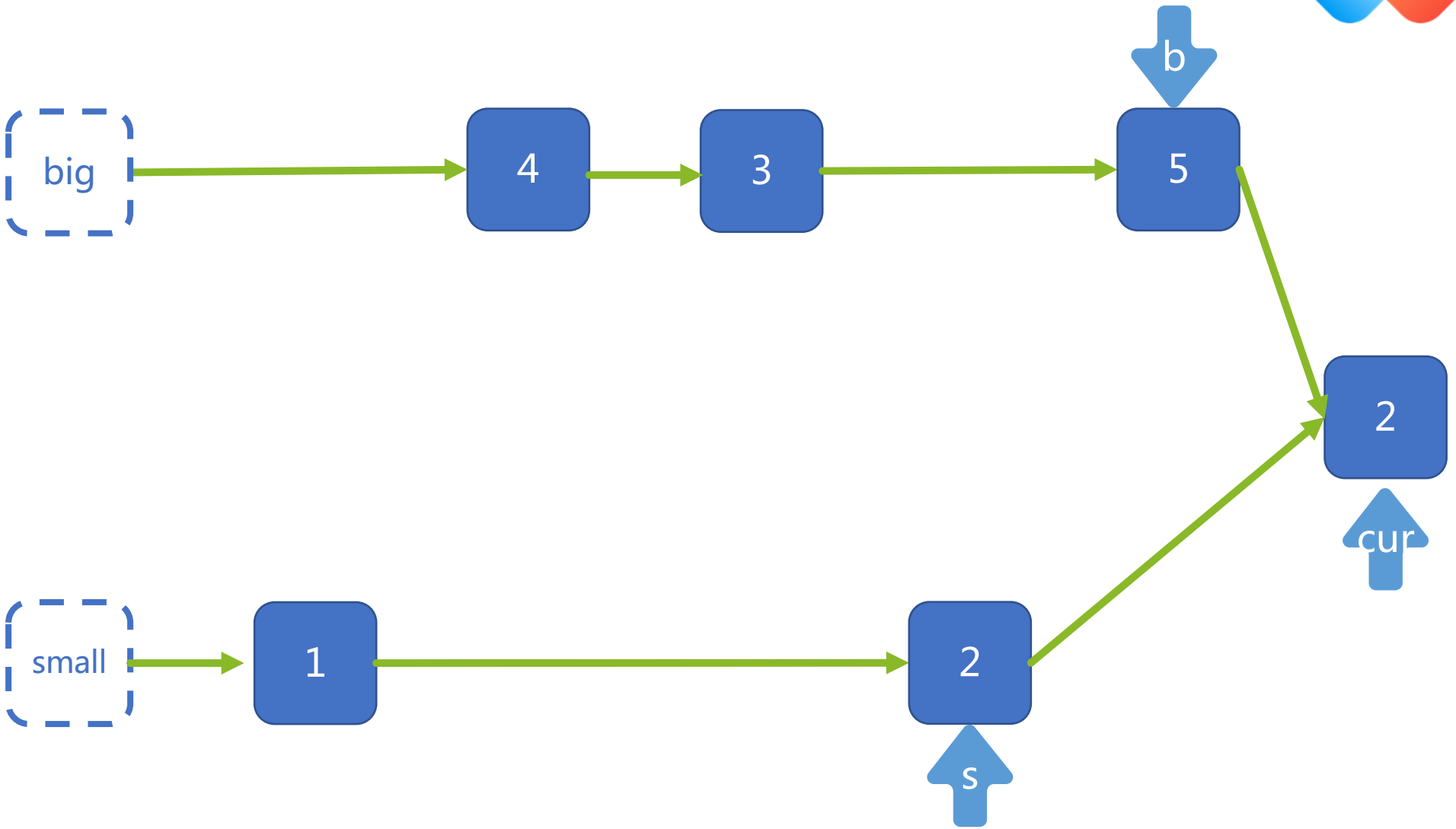
LeetCode-86 分隔链表



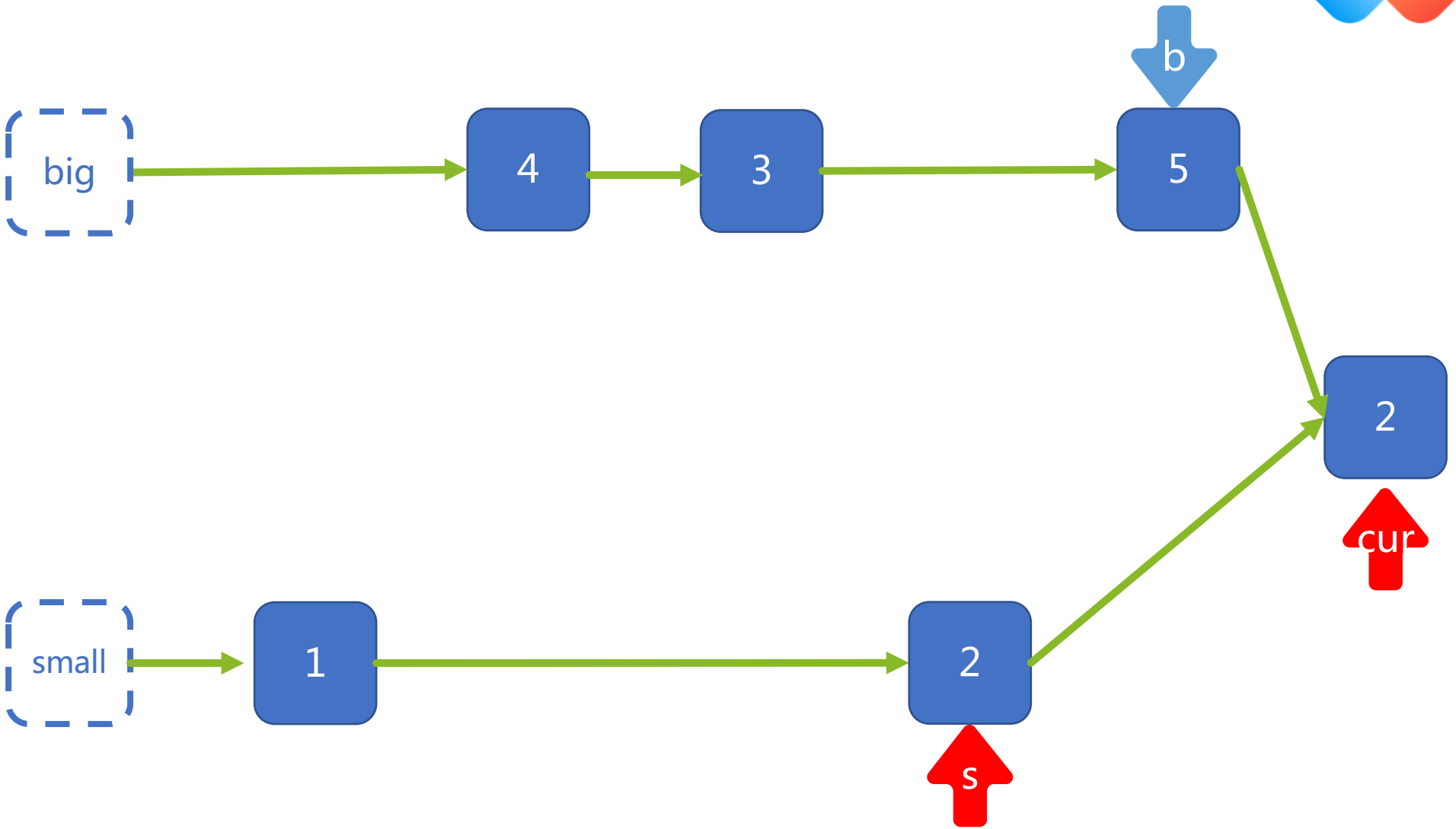
LeetCode-86 分隔链表



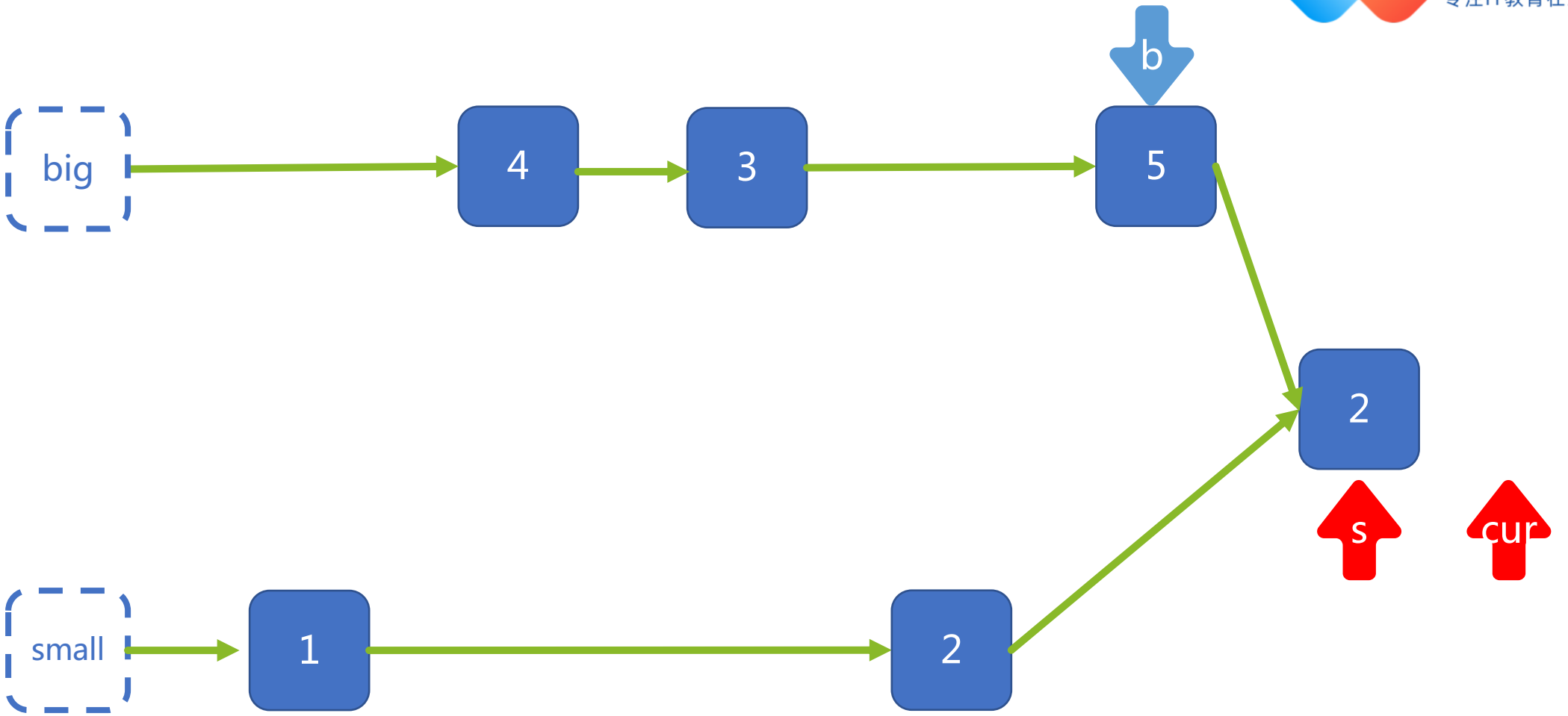
LeetCode-86 分隔链表



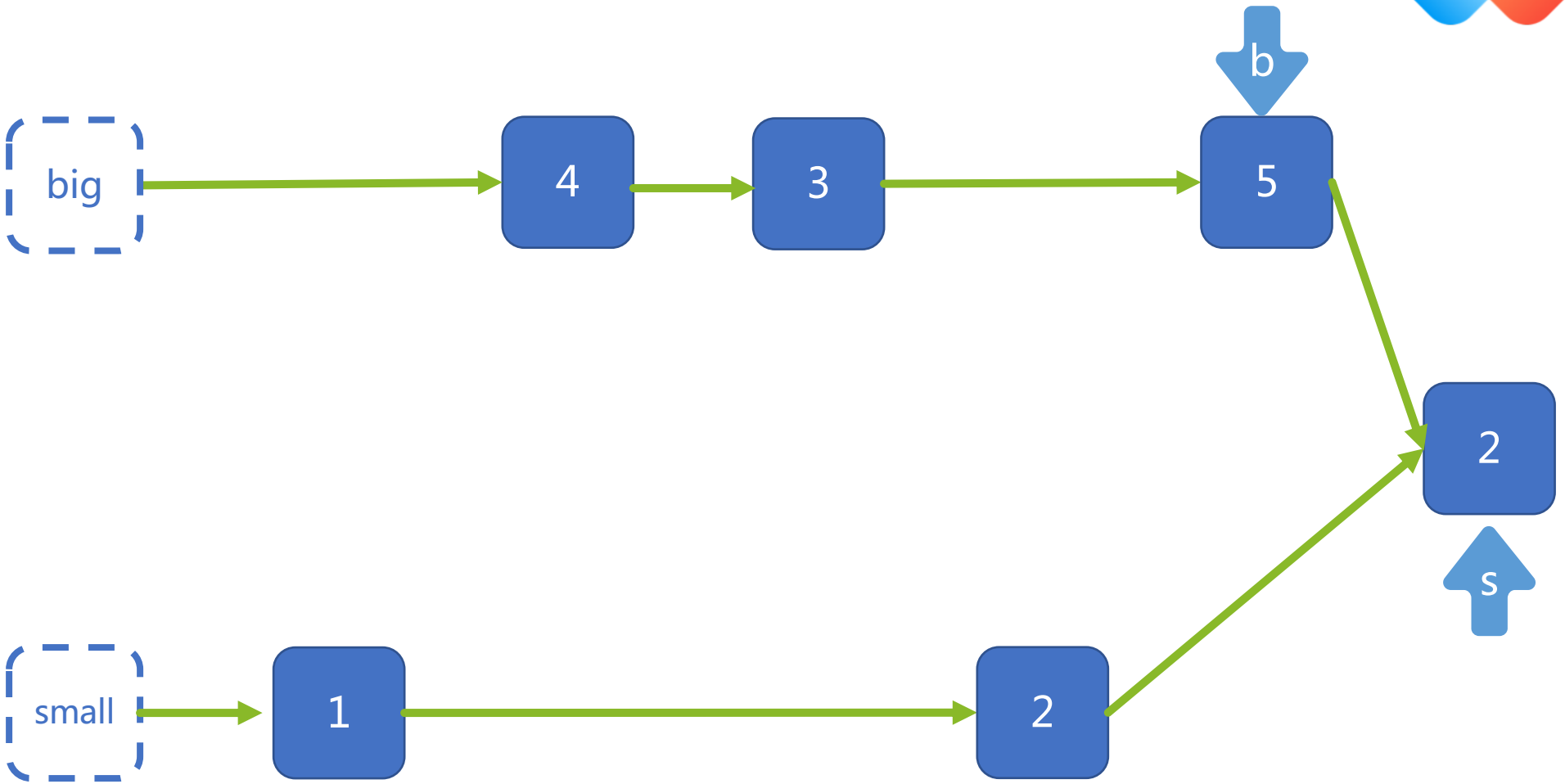
LeetCode-86 分隔链表



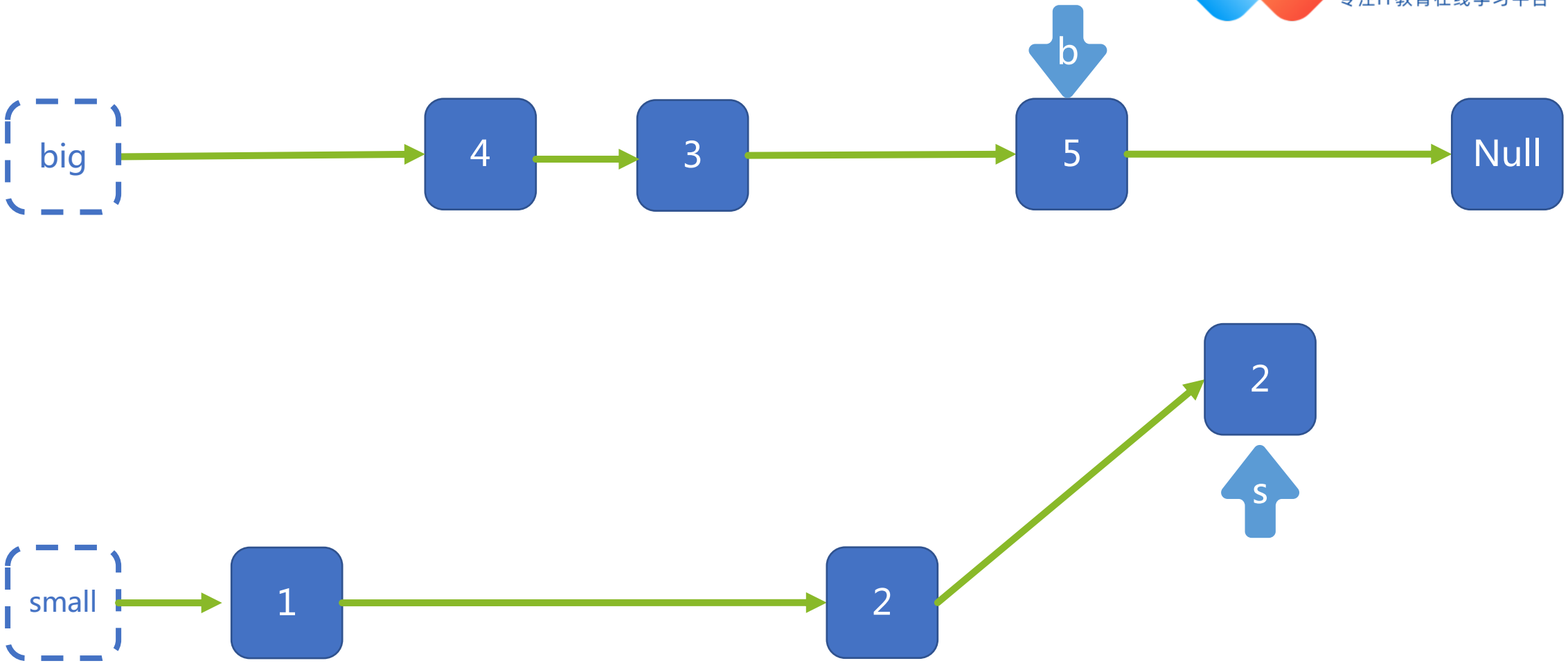
LeetCode-86 分隔链表



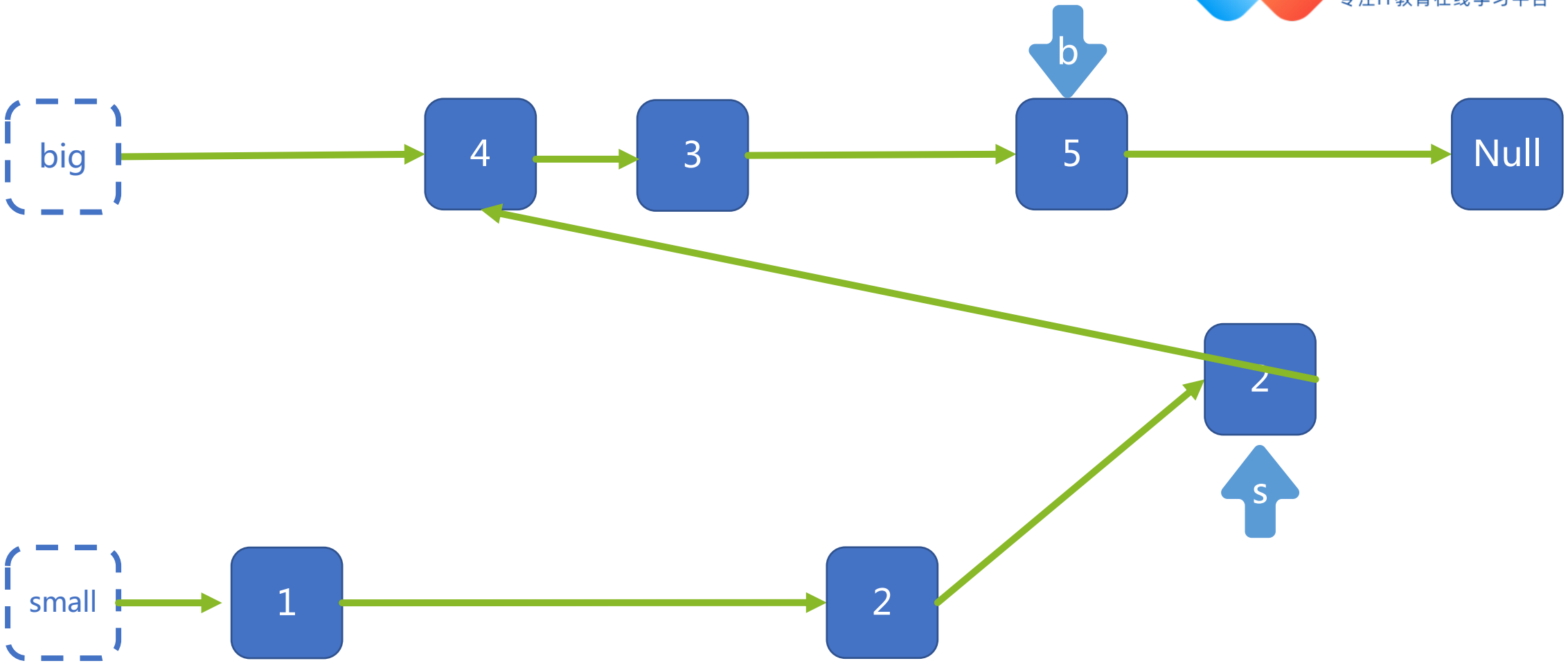
LeetCode-86 分隔链表



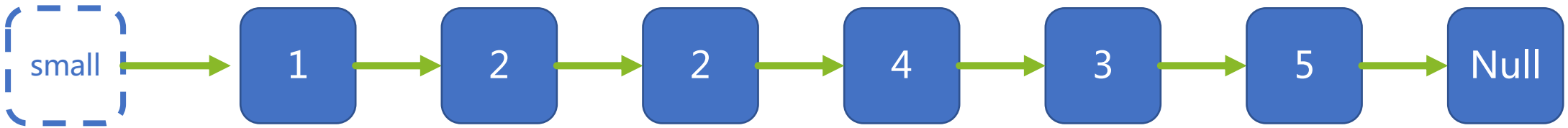
LeetCode-86 分隔链表



LeetCode-86 分隔链表



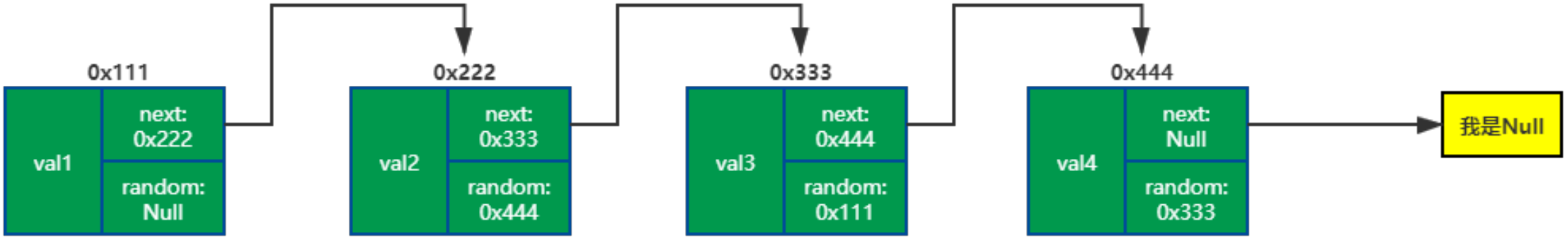
LeetCode-86 分隔链表



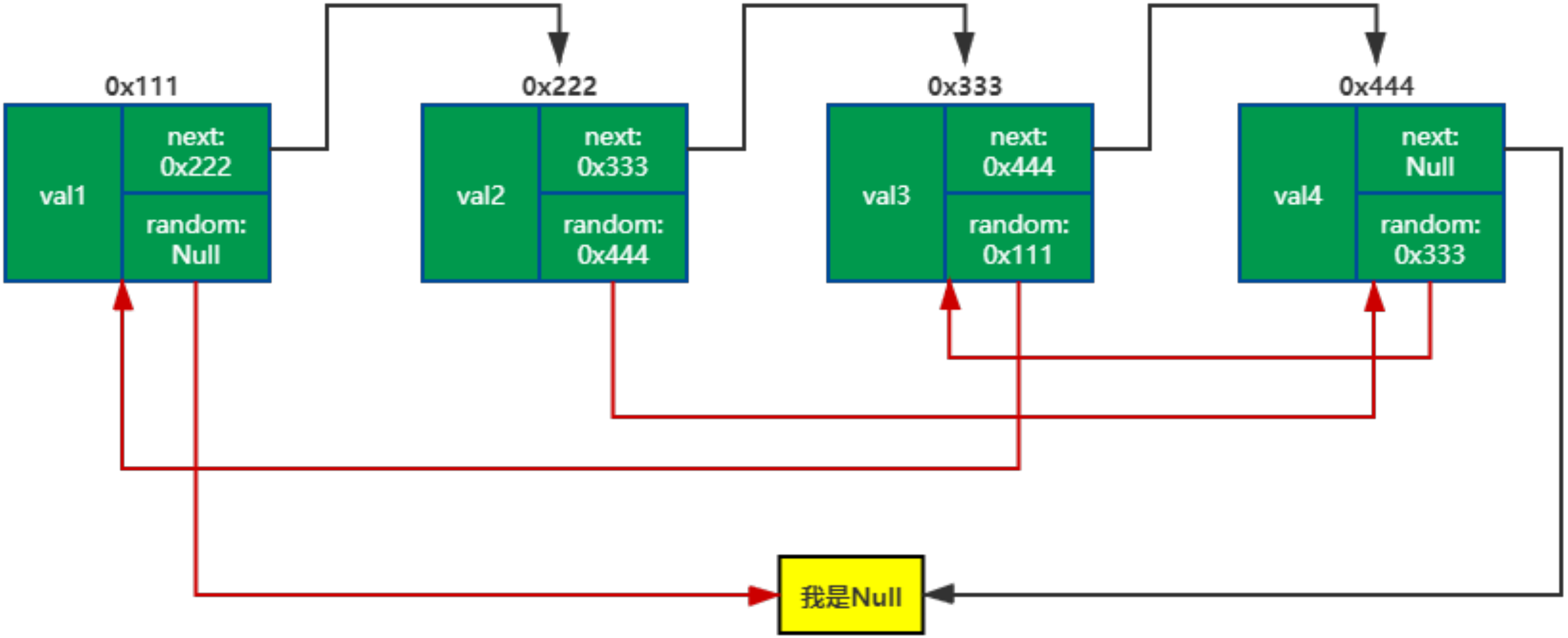
138.复制带随机指针的链表

门徒计划，带你开启算法精进之路

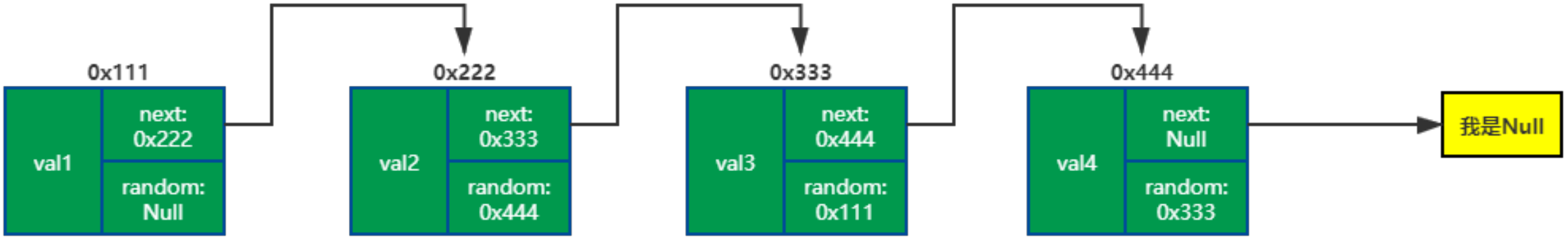
LeetCode-138 复制带随机指针的链表



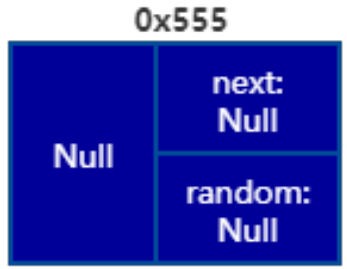
LeetCode-138 复制带随机指针的链表



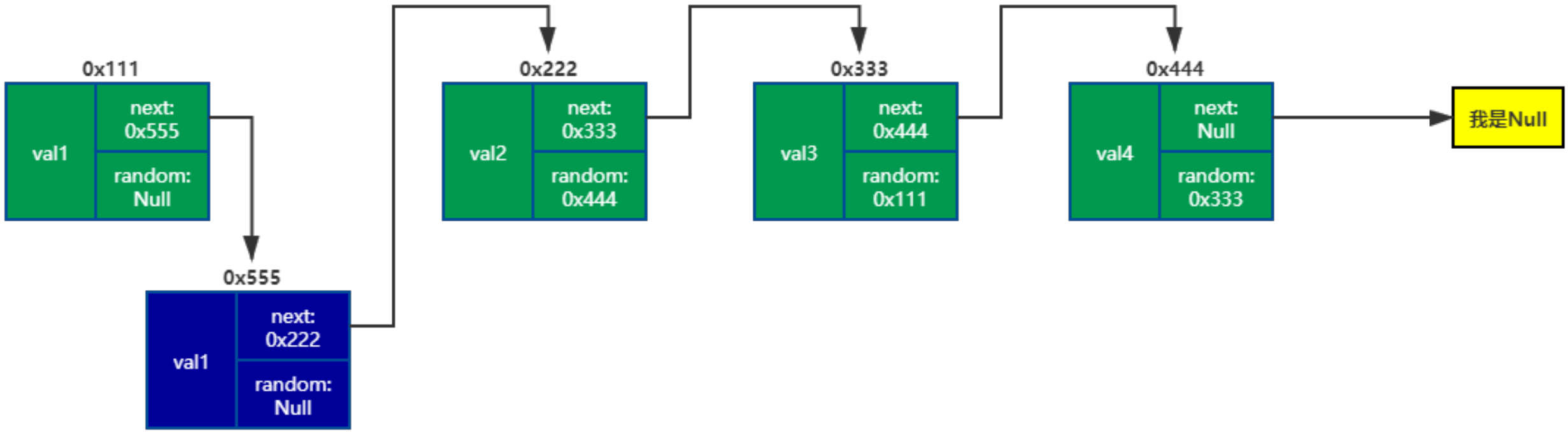
LeetCode-138 复制带随机指针的链表



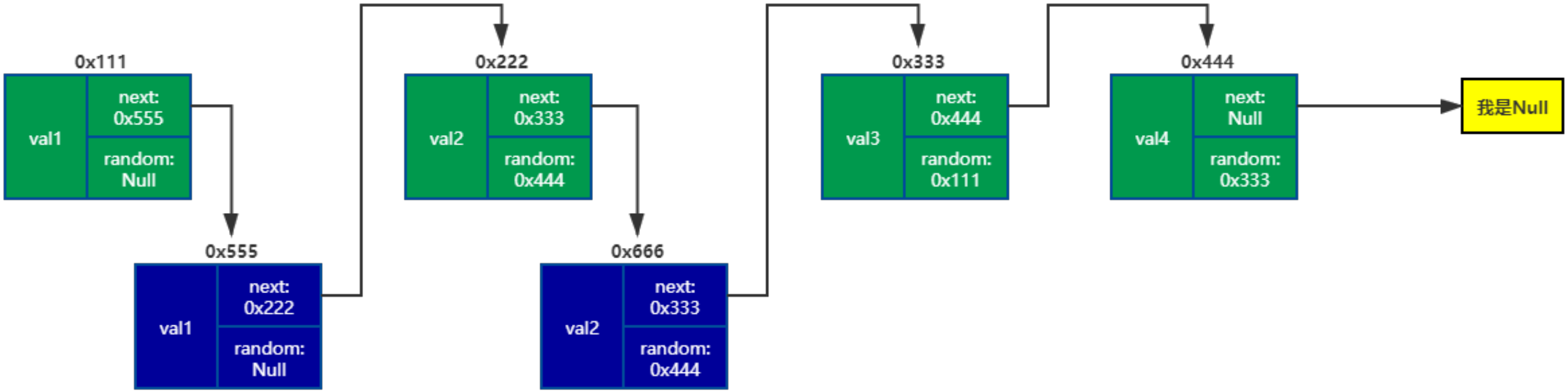
LeetCode-138 复制带随机指针的链表



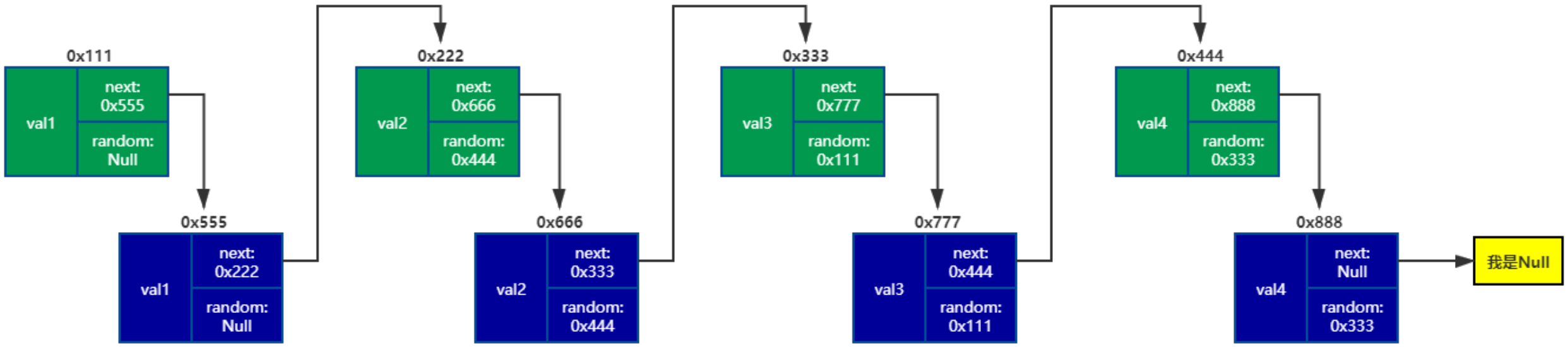
LeetCode-138 复制带随机指针的链表



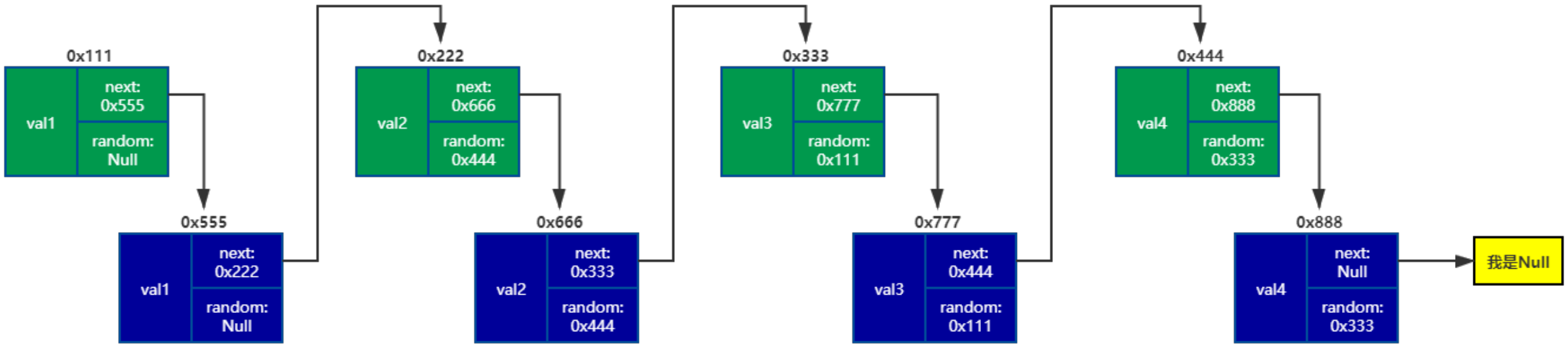
LeetCode-138 复制带随机指针的链表



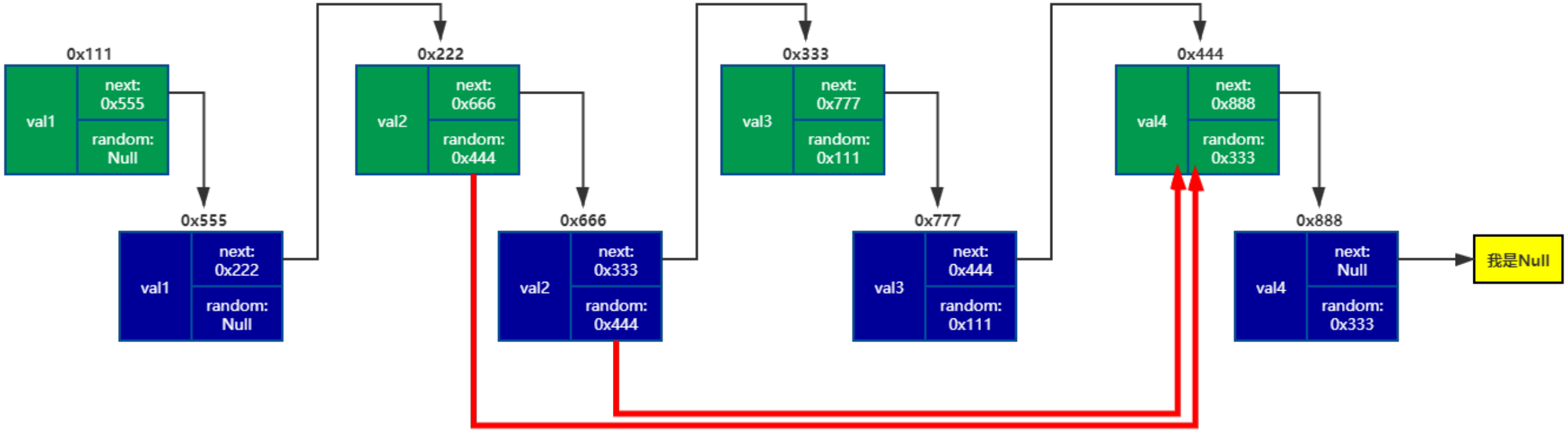
LeetCode-138 复制带随机指针的链表



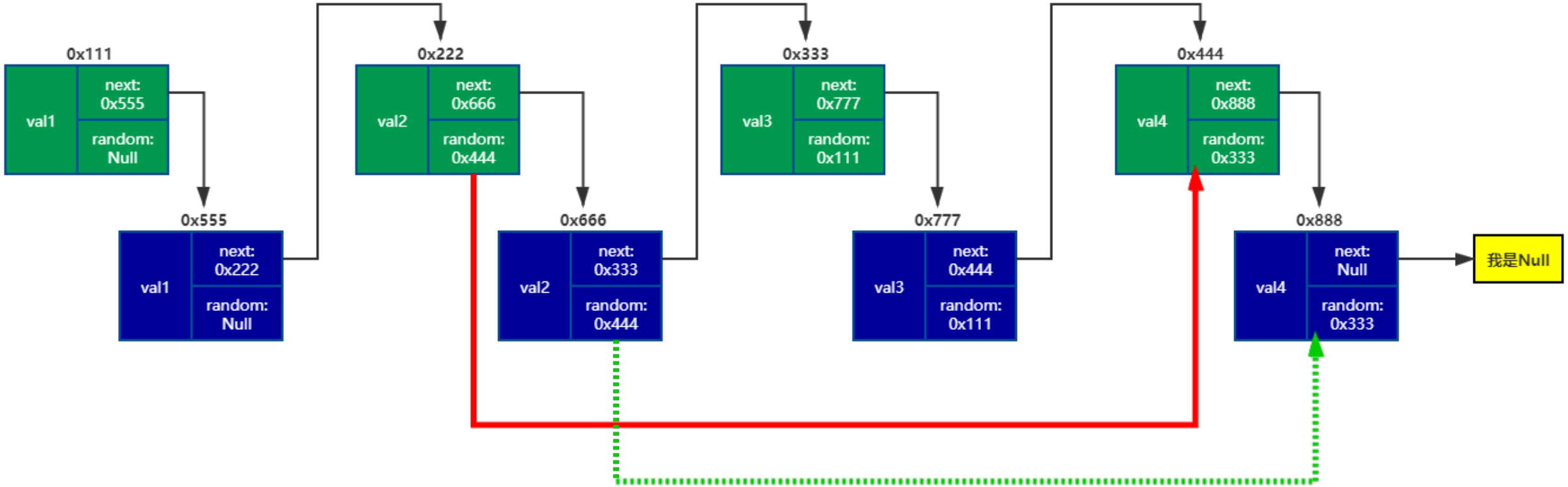
LeetCode-138 复制带随机指针的链表



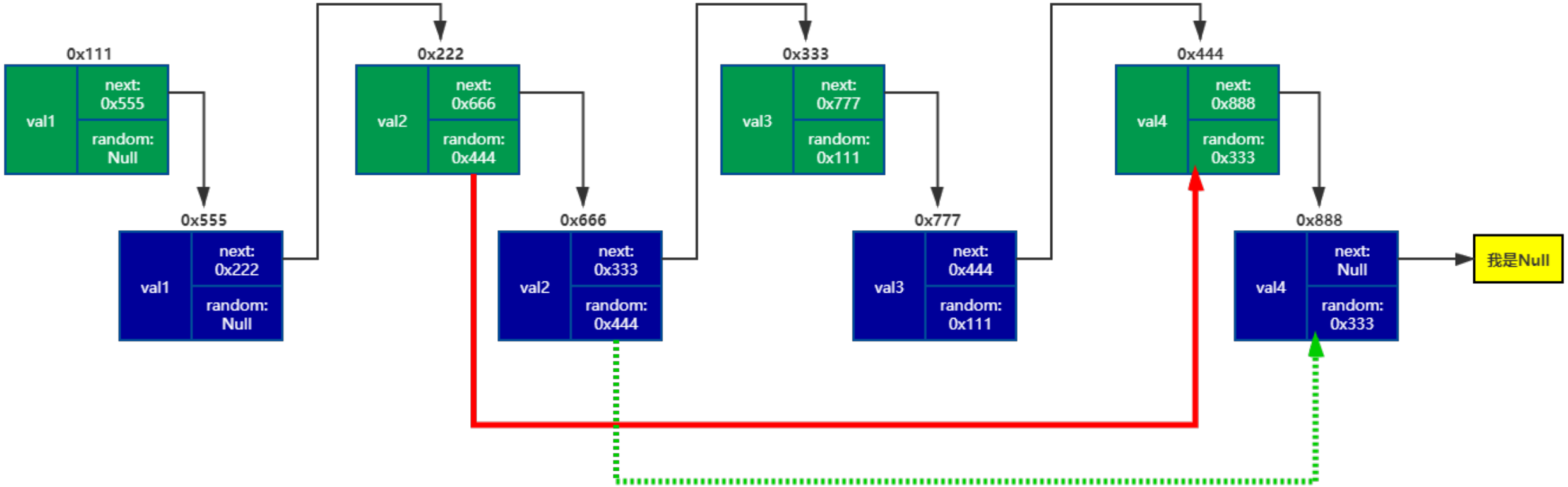
LeetCode-138 复制带随机指针的链表



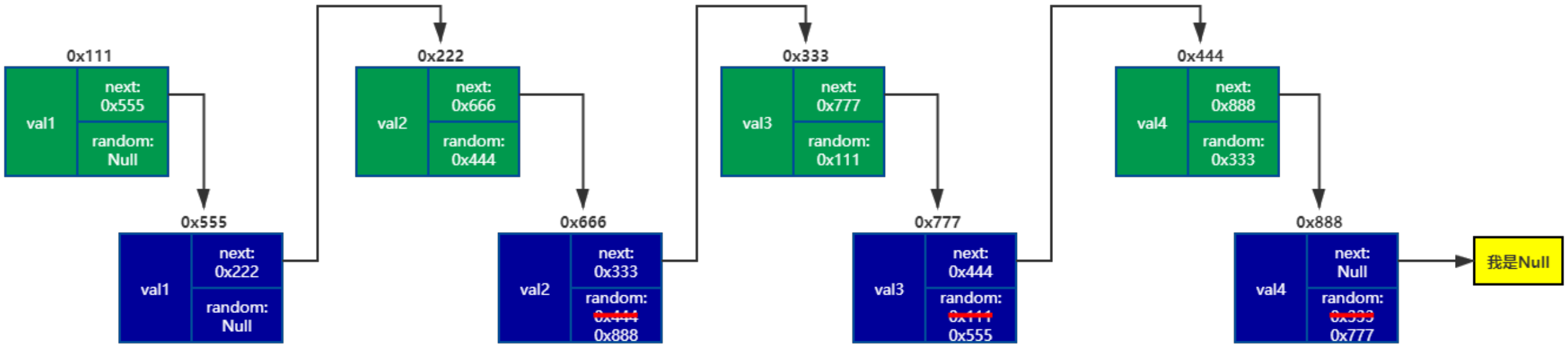
LeetCode-138 复制带随机指针的链表



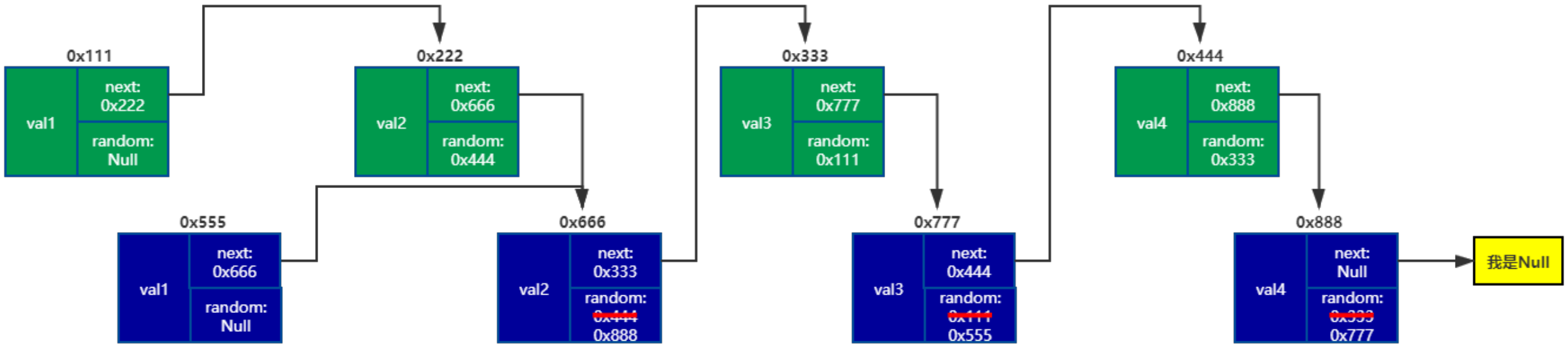
LeetCode-138 复制带随机指针的链表



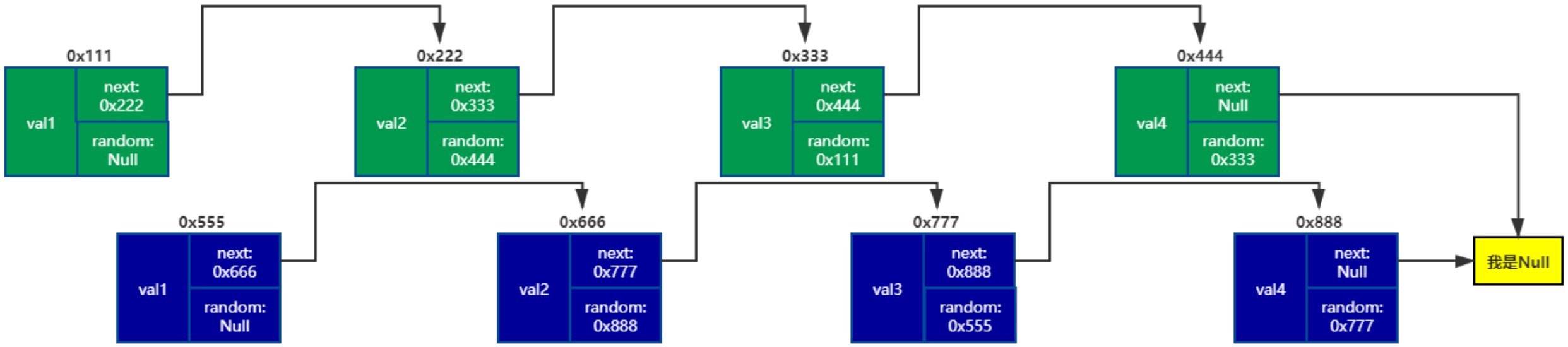
LeetCode-138 复制带随机指针的链表



LeetCode-138 复制带随机指针的链表



LeetCode-138 复制带随机指针的链表



经典面试题-队列的封装与使用

大约用时：（ 75 mins ）

下一部分：经典面试题-智力发散题

622.设计循环队列

门徒计划，带你开启算法精进之路

LeetCode-622 设计循环队列

641.设计双向循环队列

门徒计划，带你开启算法精进之路

LeetCode-641 设计双向循环队列

1670.设计前中后队列

门徒计划，带你开启算法精进之路

LeetCode-1670 设计前中后队列

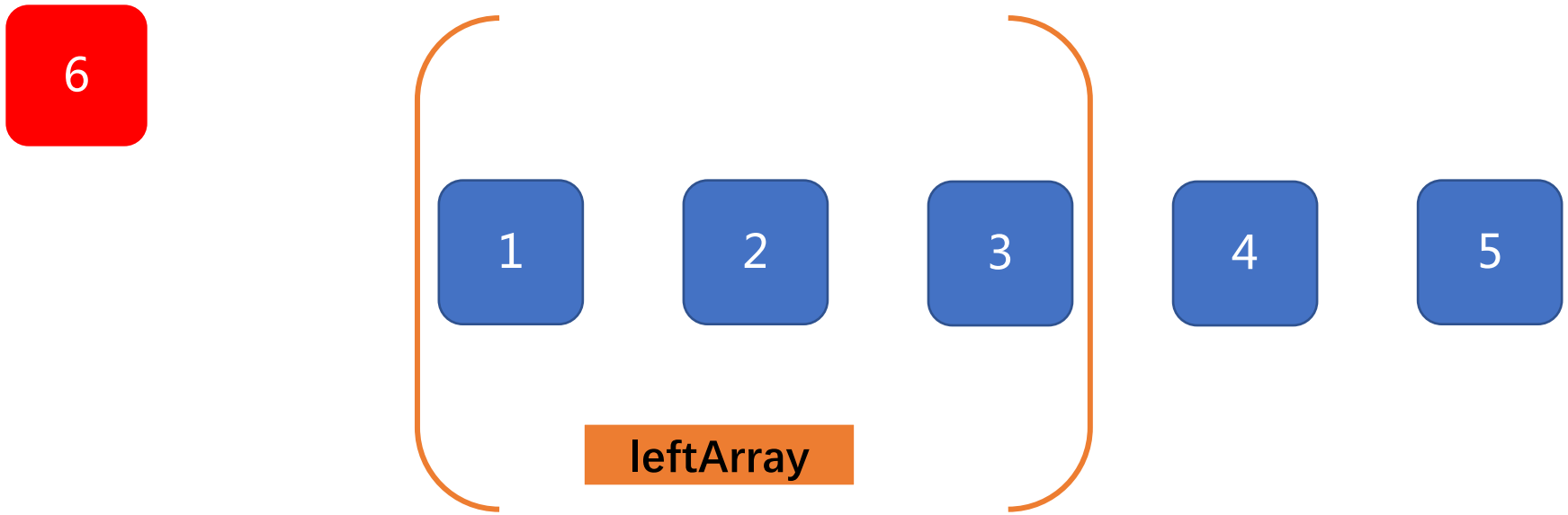
举例，我们有上面这样的 一个队列

LeetCode-1670 设计前中后队列

第一个要求，将 6 添加到队列的最前面

LeetCode-1670 设计前中后队列

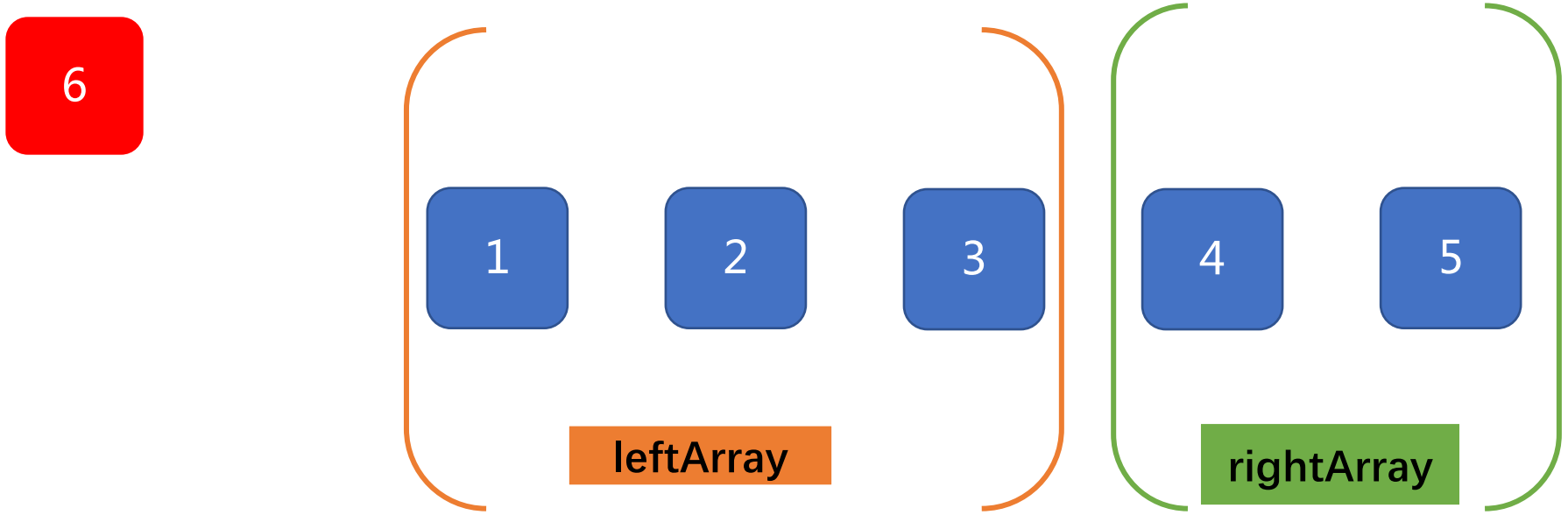
第一个要求，将 6 添加到队列的最前面



将队列分为左右两个队列，左队列 leftArray 用黄括号括起来

LeetCode-1670 设计前中后队列

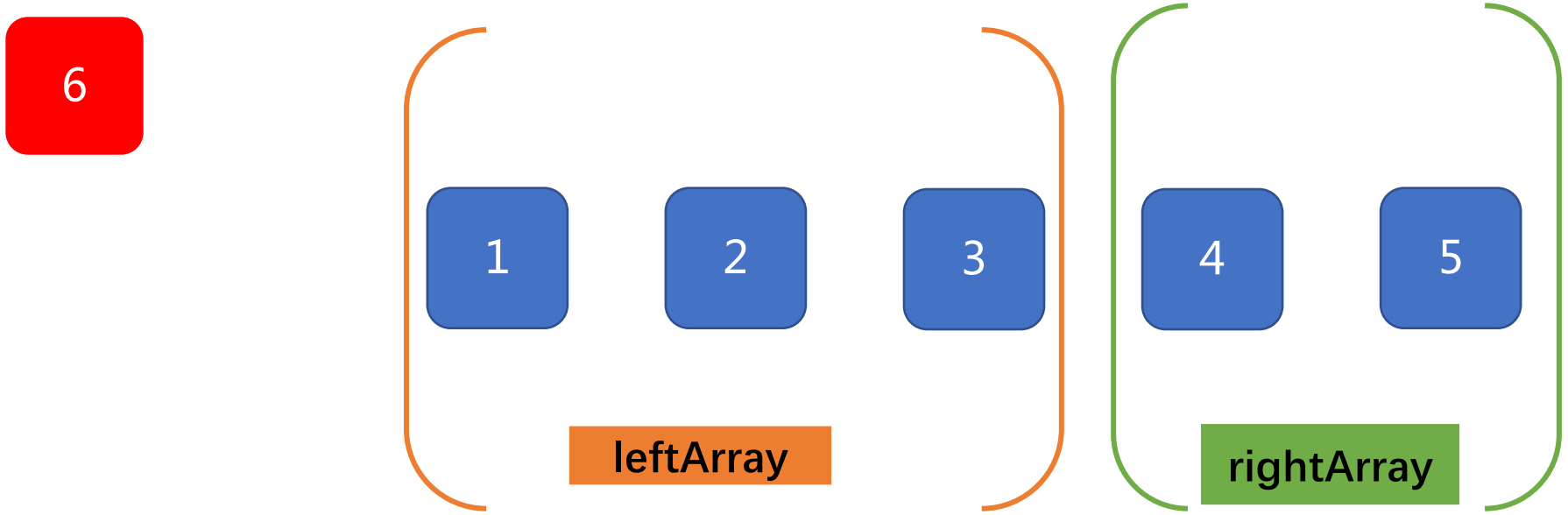
第一个要求，将 6 添加到队列的最前面



右队列 rightArray 用绿括号括起来

LeetCode-1670 设计前中后队列

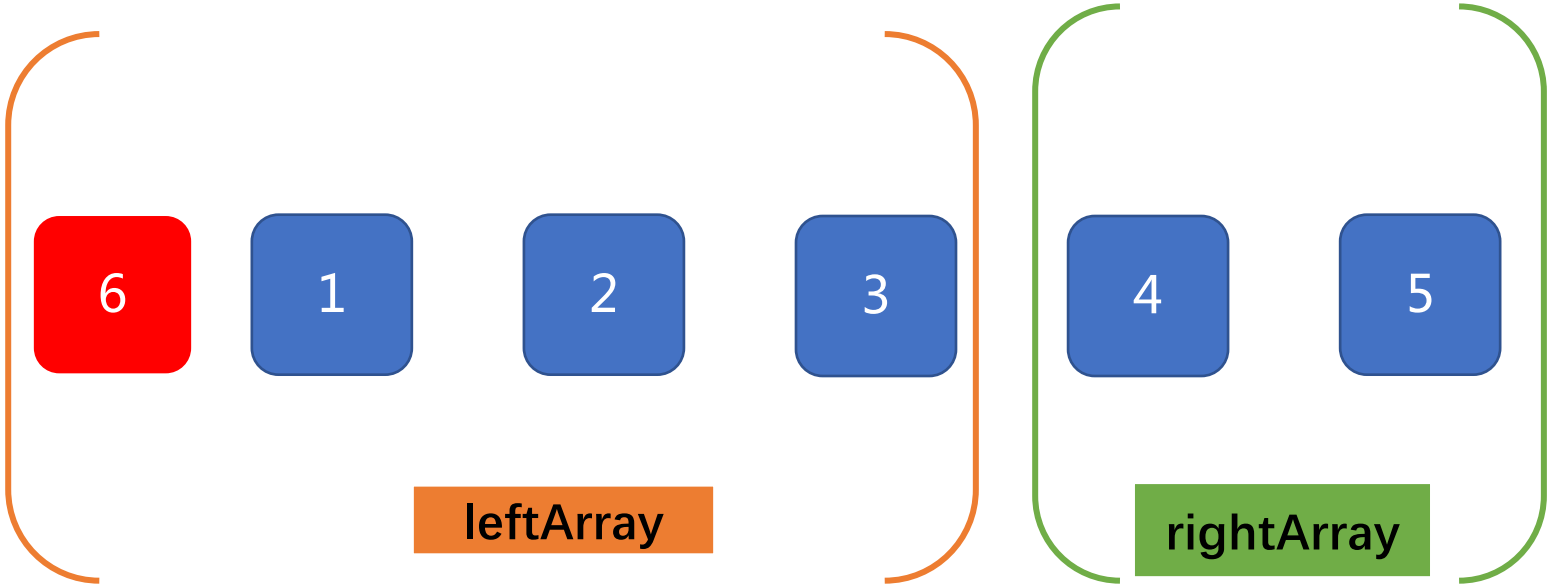
第一个要求，将 6 添加到队列的最前面



让左队列在第一位新增一个数据，用到了方法unshift

LeetCode-1670 设计前中后队列

第一个要求，将 6 添加到队列的最前面

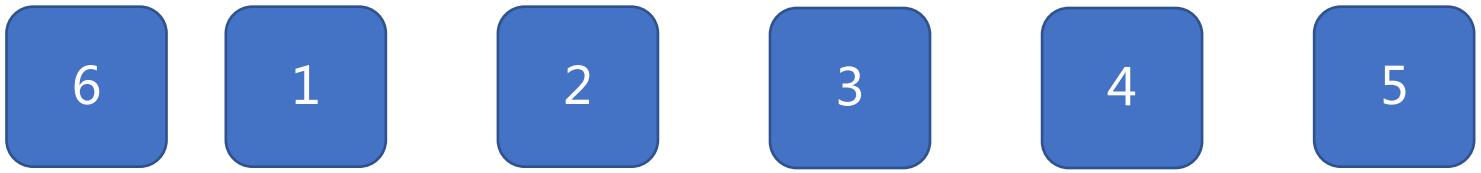


添加后就变成了上面的效果

LeetCode-1670 设计前中后队列

海贼宝藏
专注IT教育在线学习平台

第一个要求，将 6 添加到队列的最前面



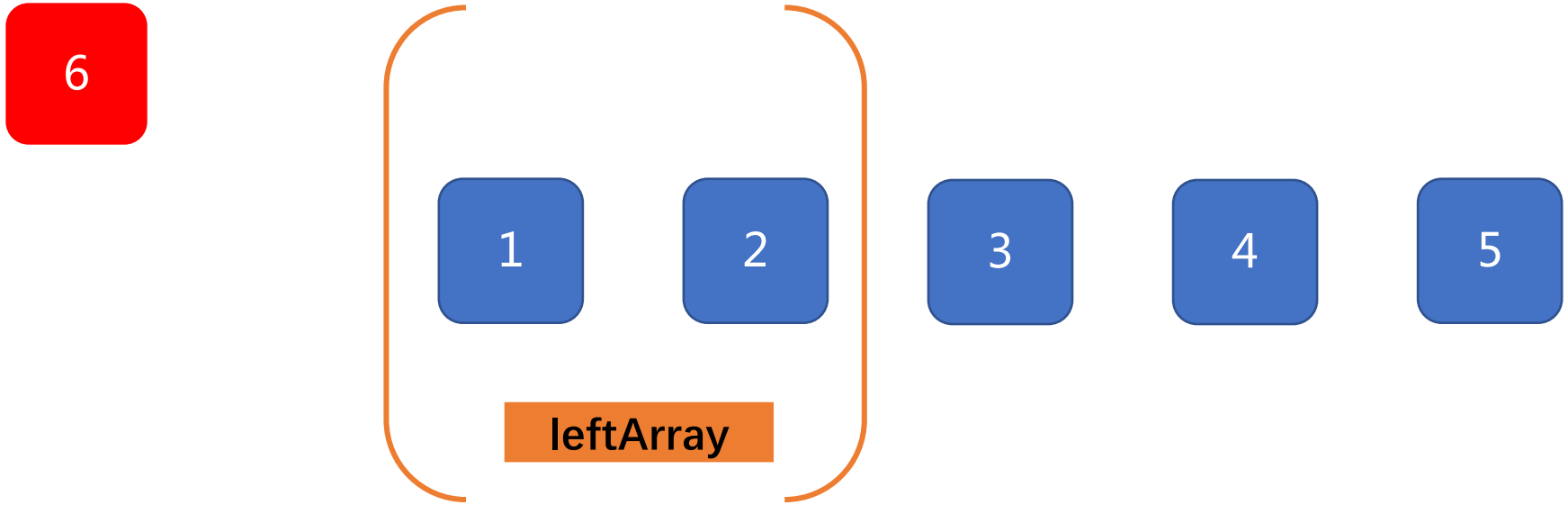
这便是在队列的 最前面 新增一位最后的效果

LeetCode-1670 设计前中后队列

第二个要求，将 6 添加到队列的最中间

LeetCode-1670 设计前中后队列

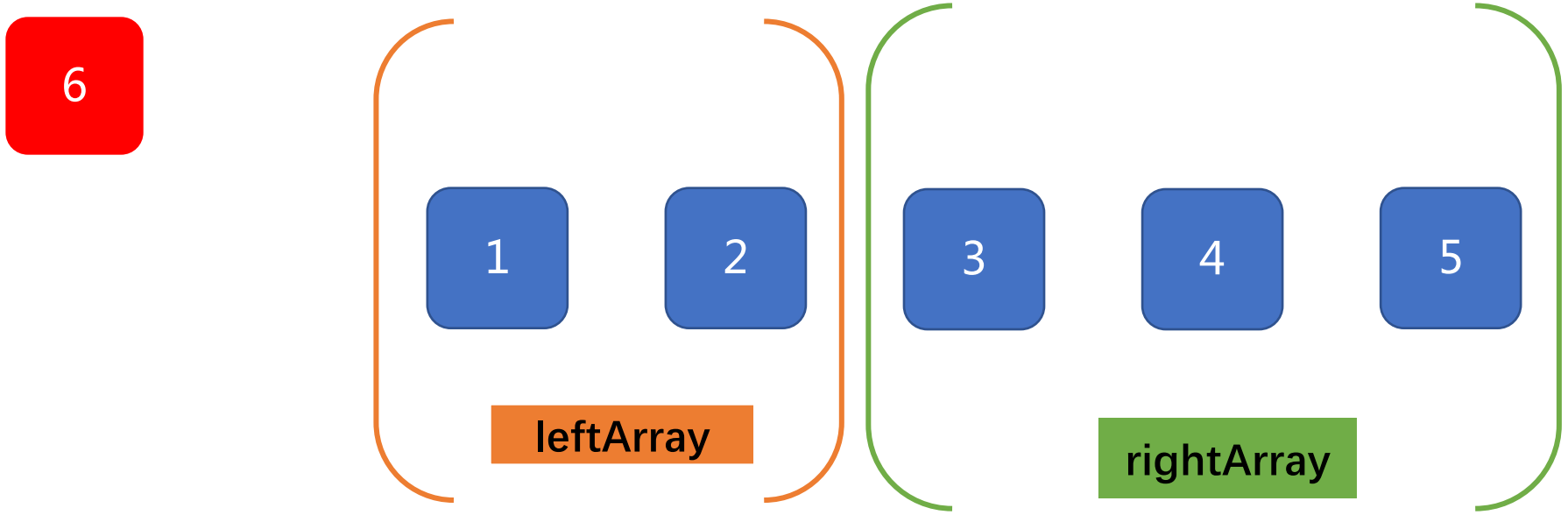
第二个要求，将 6 添加到队列的最中间



将队列分为左右两个队列，左队列 leftArray 用黄括号括起来

LeetCode-1670 设计前中后队列

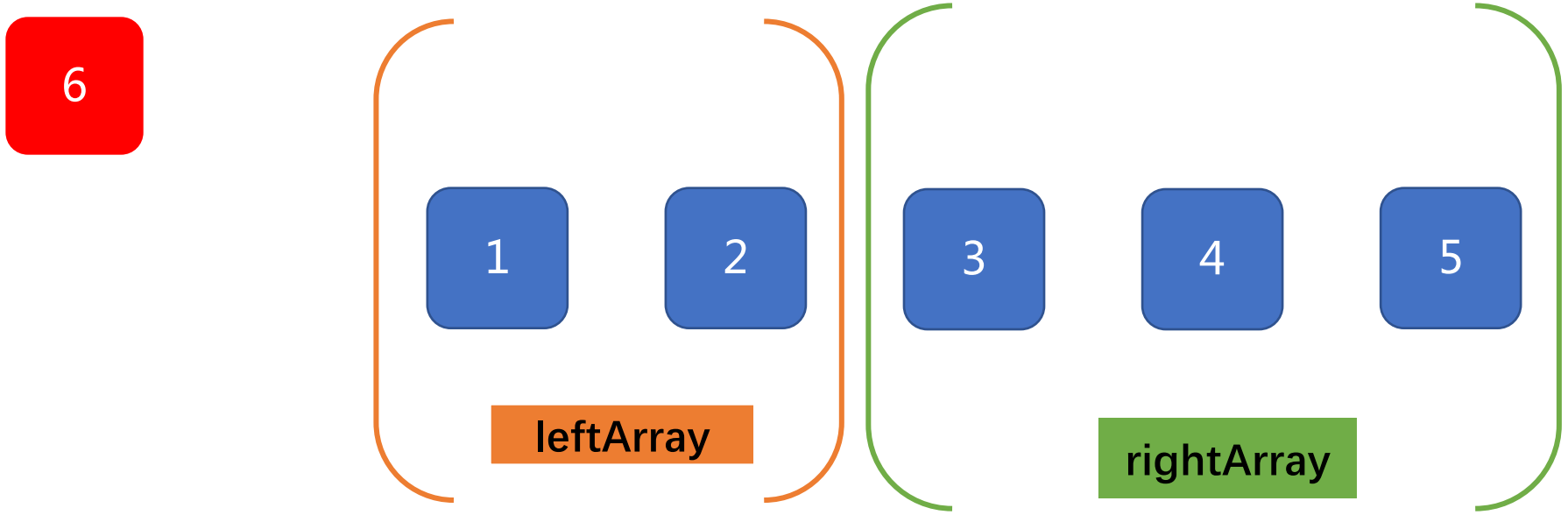
第二个要求，将 6 添加到队列的最中间



右队列 rightArray 用绿括号括起来

LeetCode-1670 设计前中后队列

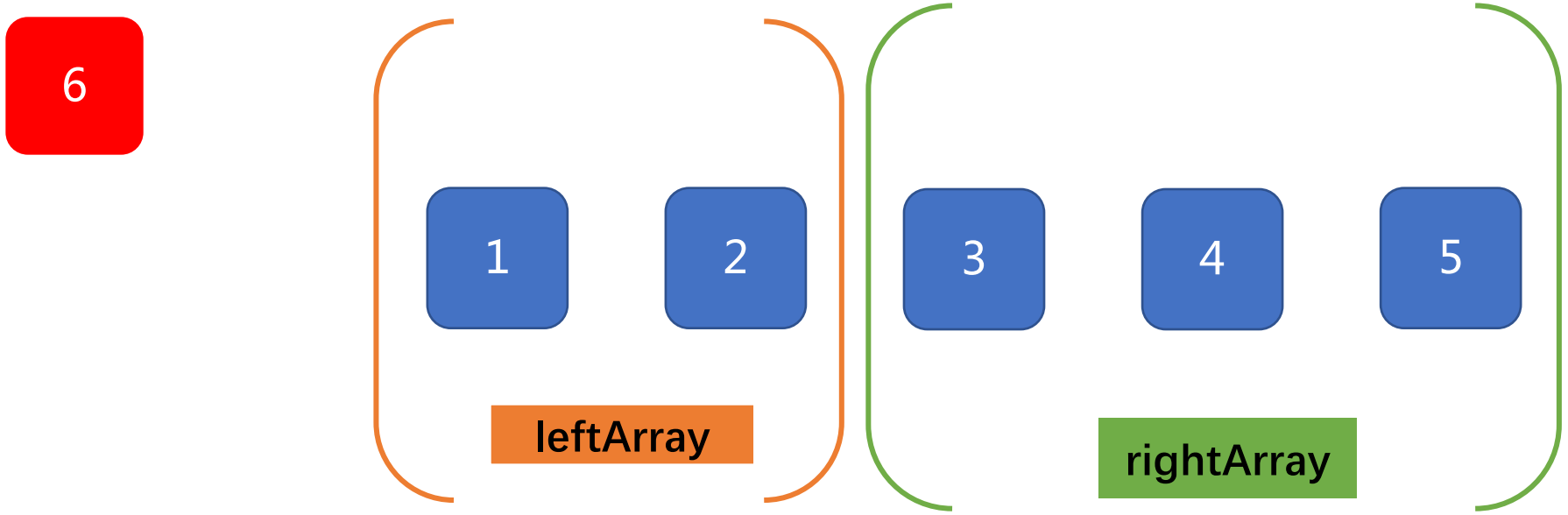
第二个要求，将 6 添加到队列的最中间



如果两个队列长度不等 leftArray.length != rightArray.length，

LeetCode-1670 设计前中后队列

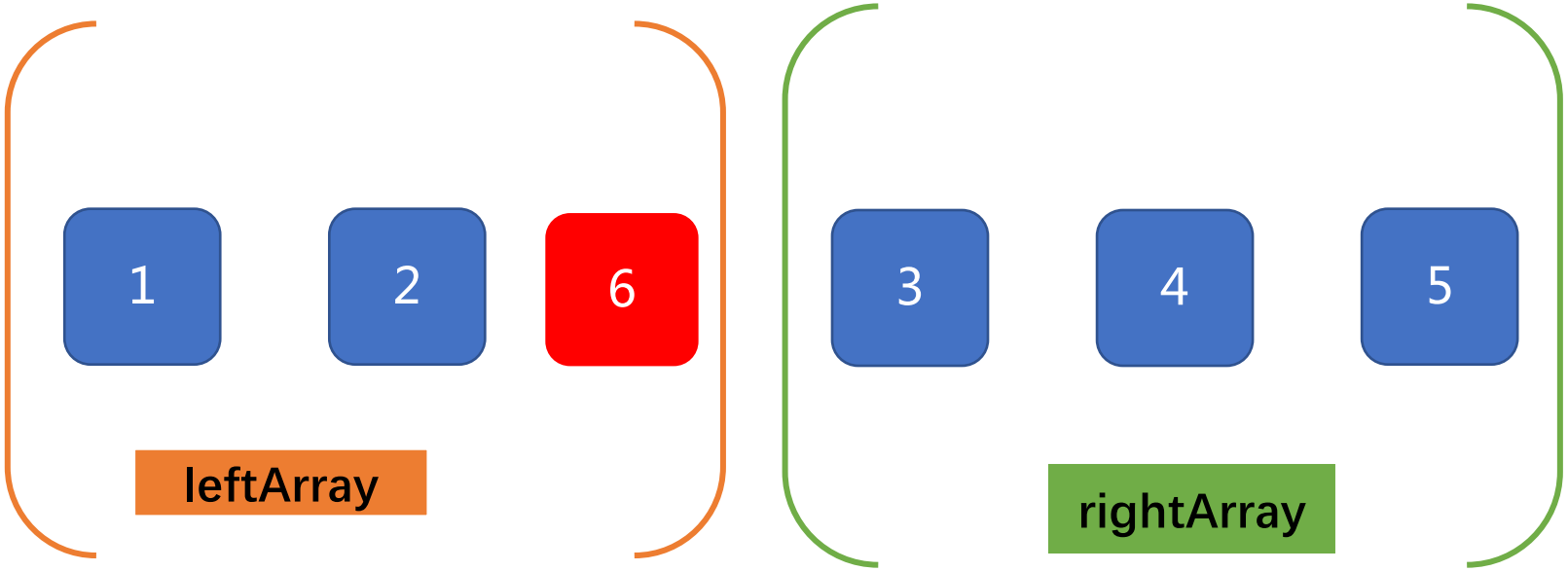
第二个要求，将 6 添加到队列的最中间



就让左队列再最后一位新增一个数据，用到了方法push

LeetCode-1670 设计前中后队列

第二个要求，将 6 添加到队列的最中间



就让左队列再最后一位新增一个数据，用到了方法push

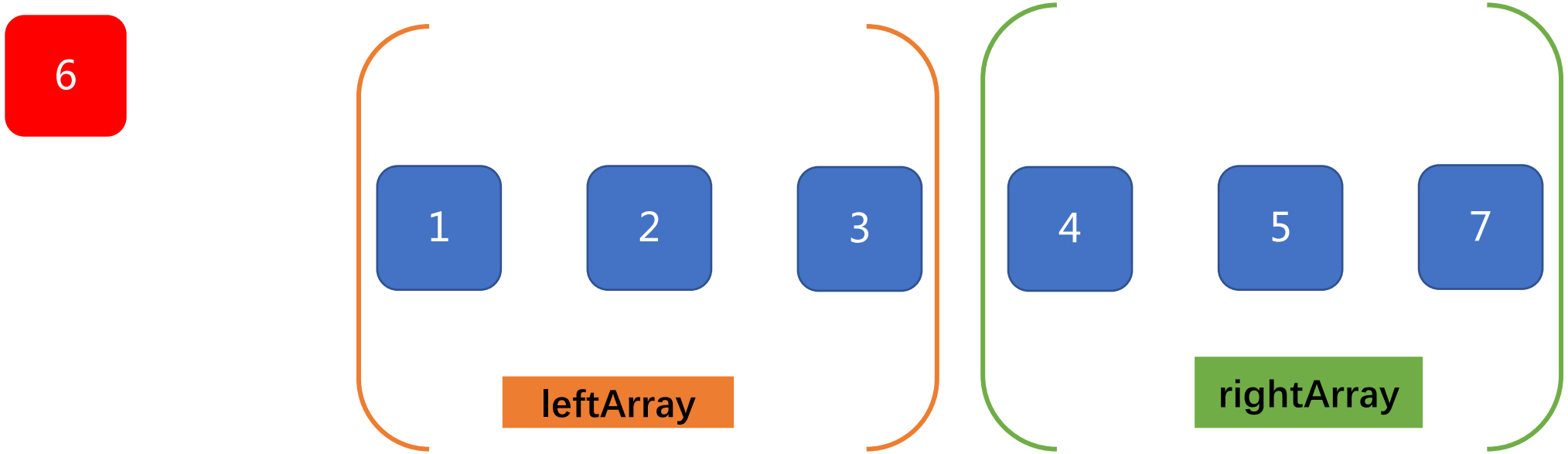
LeetCode-1670 设计前中后队列

第二个要求，将 6 添加到队列的最中间

这便是在队列正中间 新增一位最后效果

LeetCode-1670 设计前中后队列

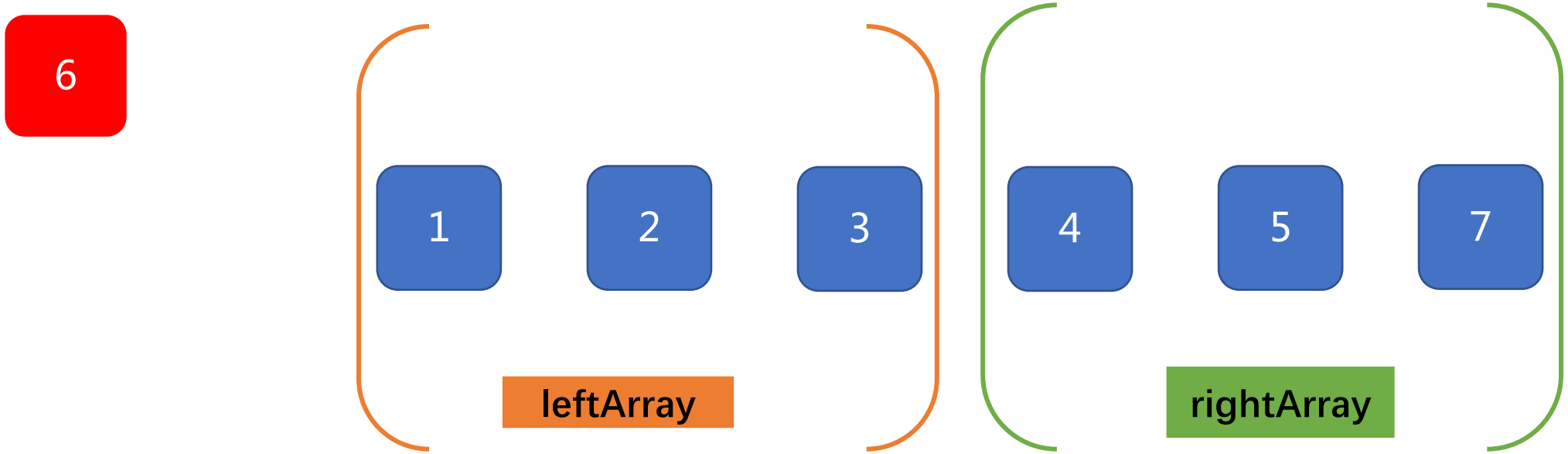
第二个要求，将 6 添加到队列的最中间



如果两个队列长度相等 leftArray.length == rightArray.length，

LeetCode-1670 设计前中后队列

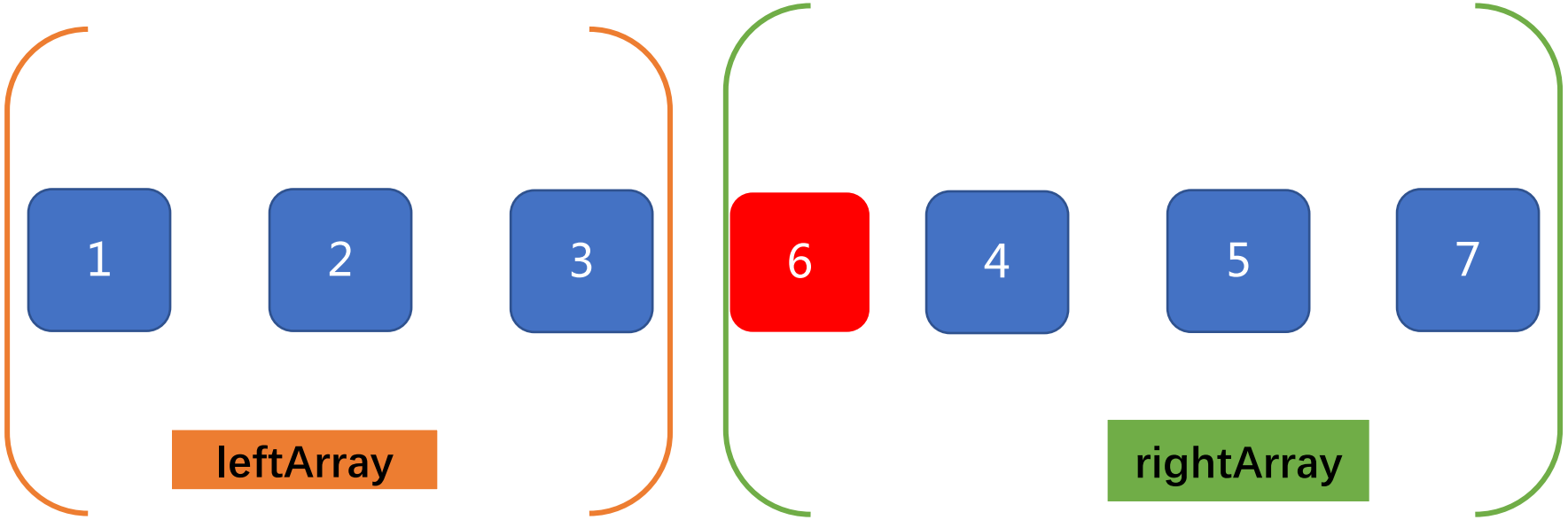
第二个要求，将 6 添加到队列的最中间



就让右队列在，第一位新增一个数据，用到了方法unshift

LeetCode-1670 设计前中后队列

第二个要求，将 6 添加到队列的最中间



就让右队列在，第一位新增一个数据，用到了方法unshift

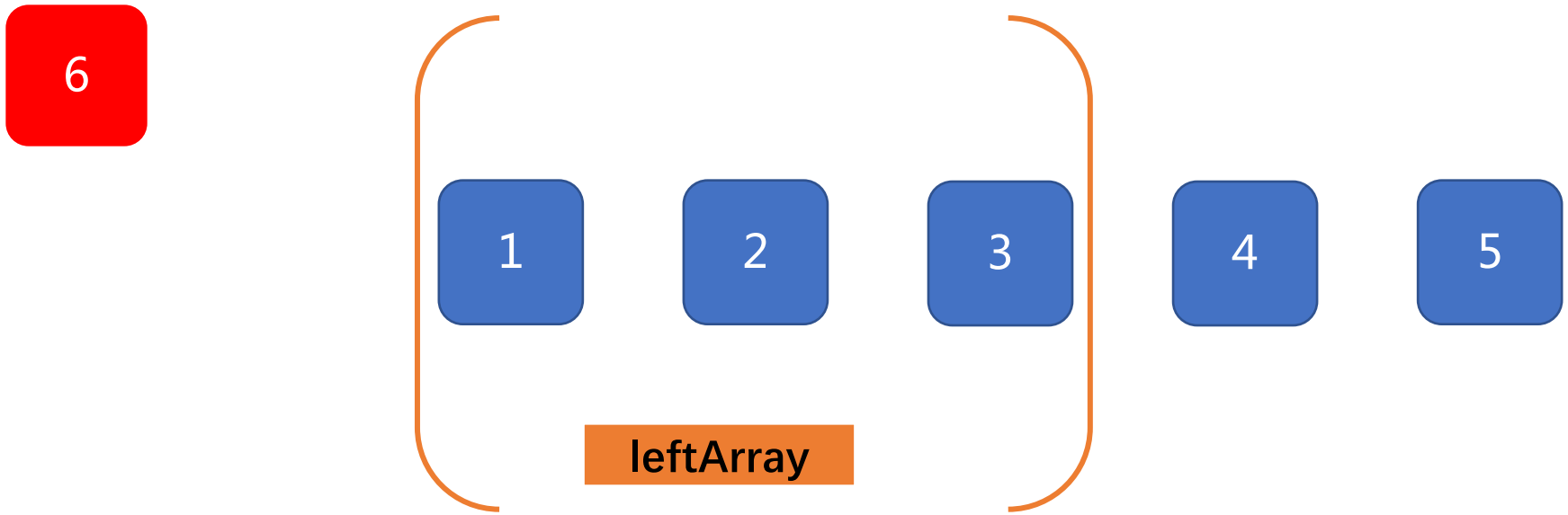
LeetCode-1670 设计前中后队列

第二个要求，将 6 添加到队列的最中间

这便是在队列正中间 新增一位最后效果

LeetCode-1670 设计前中后队列

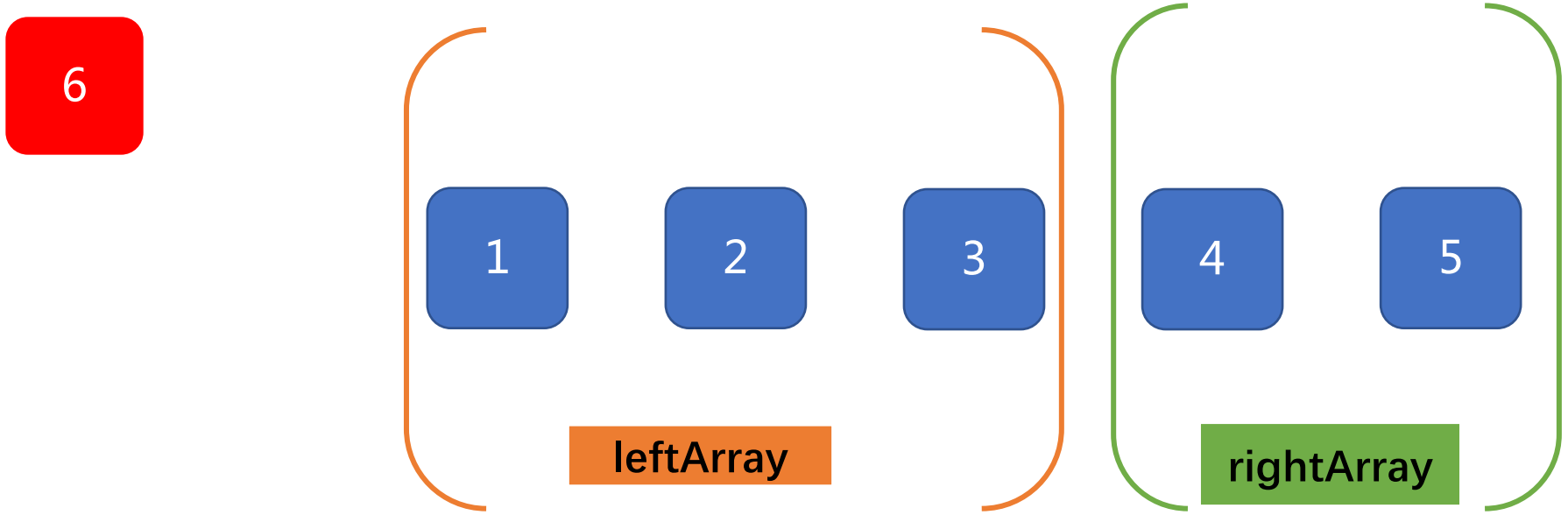
第三个要求，将 6 添加到队列的最后面



将队列分为左右两个队列，左队列 leftArray 用黄括号括起来

LeetCode-1670 设计前中后队列

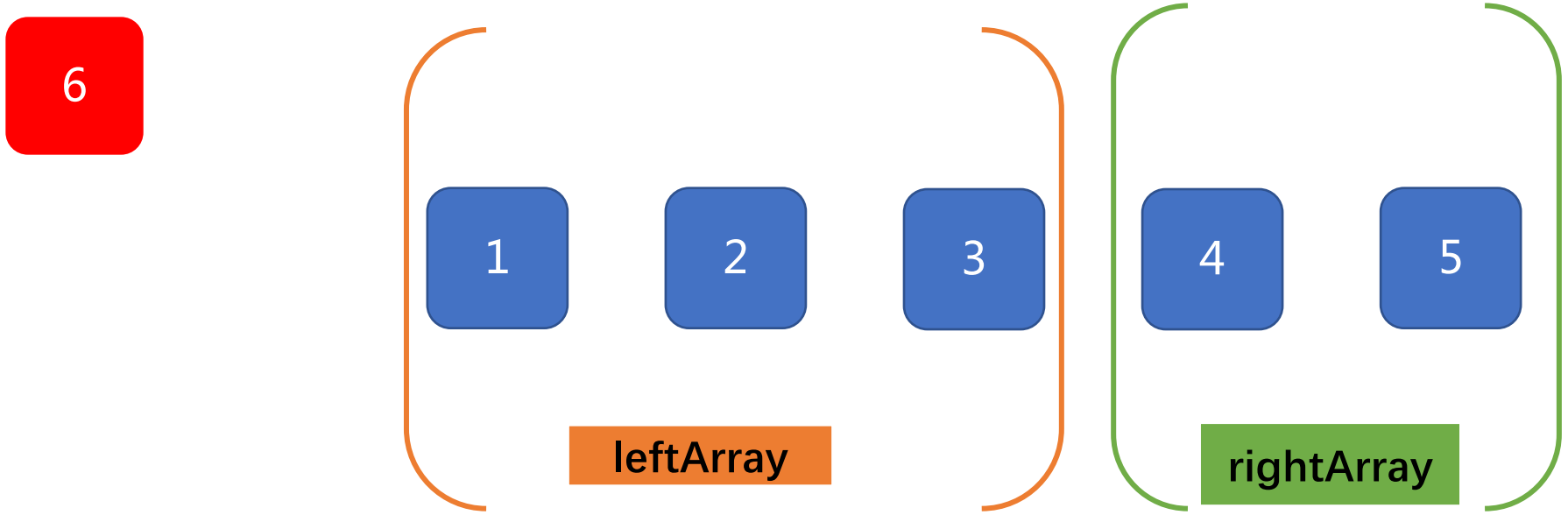
第三个要求，将 6 添加到队列的最后面



右队列 rightArray 用绿括号括起来

LeetCode-1670 设计前中后队列

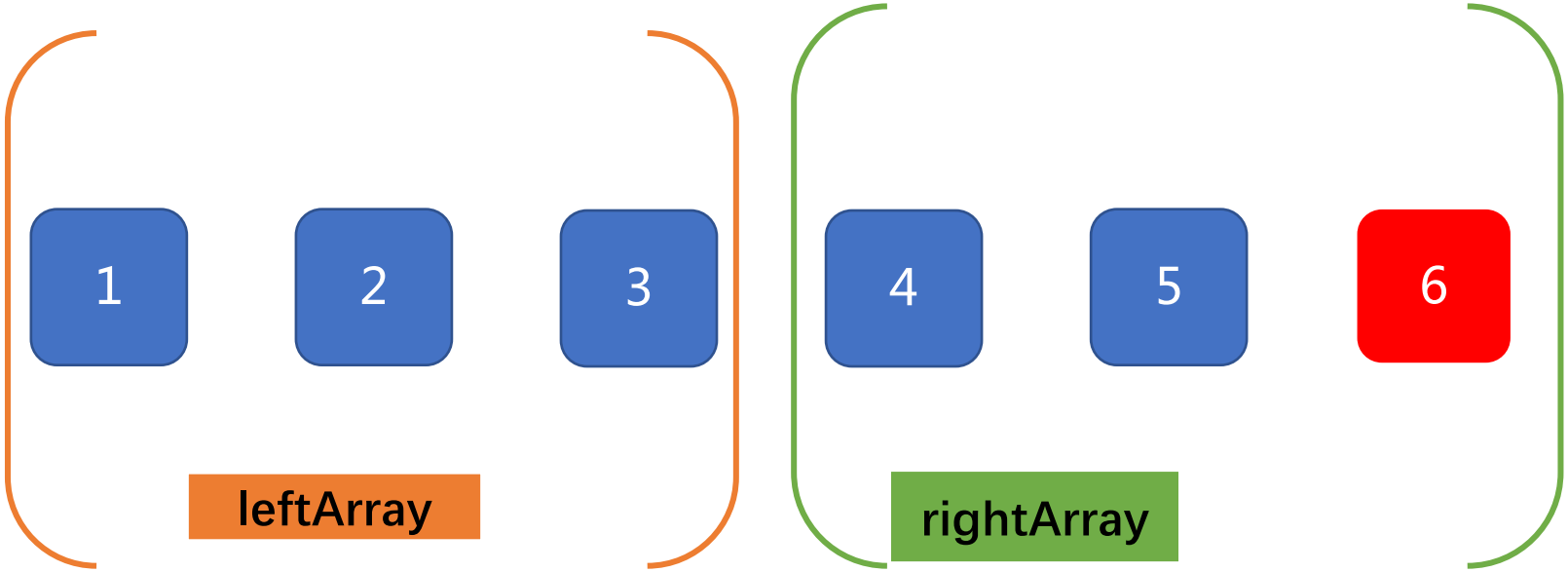
第三个要求，将 6 添加到队列的最后面



直接在右队列的最后一位新增数据，用到了方法push

LeetCode-1670 设计前中后队列

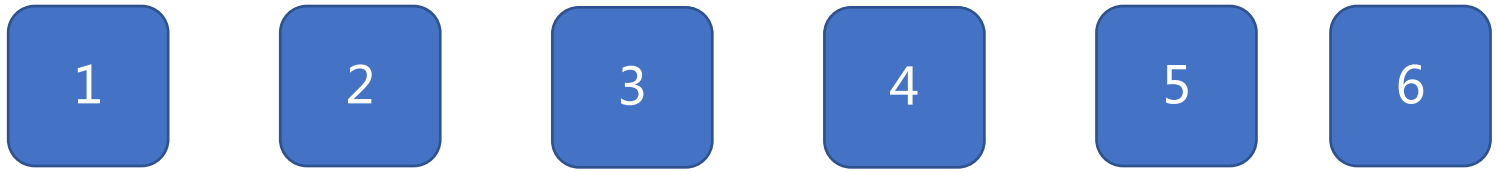
第三个要求，将 6 添加到队列的最后面



直接在右队列的最后一位新增数据，用到了方法push

LeetCode-1670 设计前中后队列

第三个要求，将 6 添加到队列的最后面



这便是在 队列的最后一位新增一位的效果

LeetCode-1670 设计前中后队列

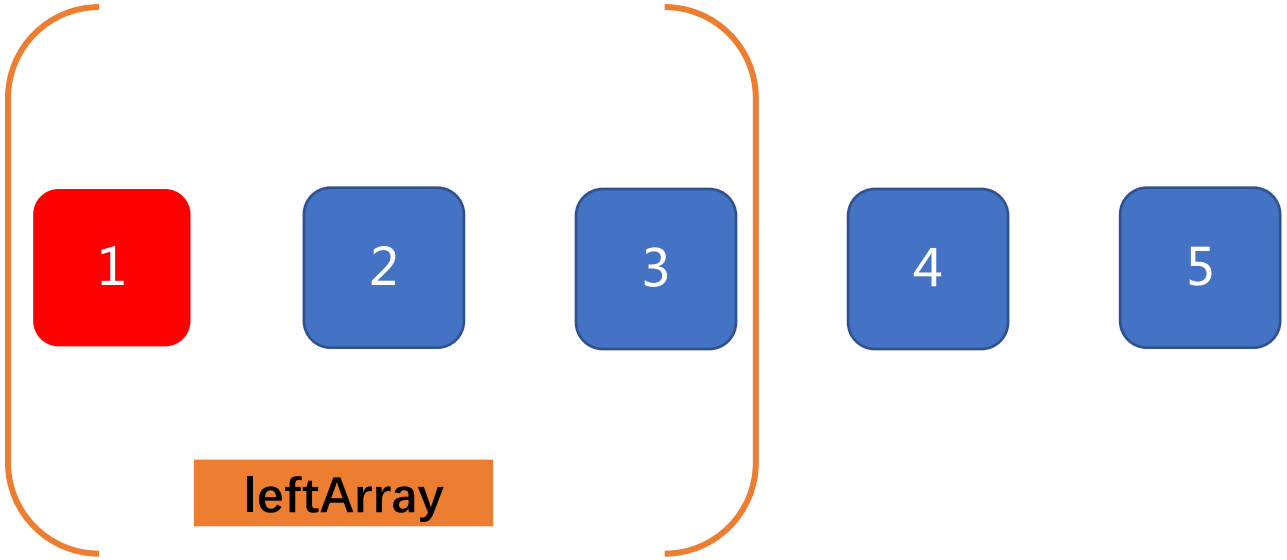
还是这个队列， 我们接下来进行 三个删除操作

LeetCode-1670 设计前中后队列

第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1

LeetCode-1670 设计前中后队列

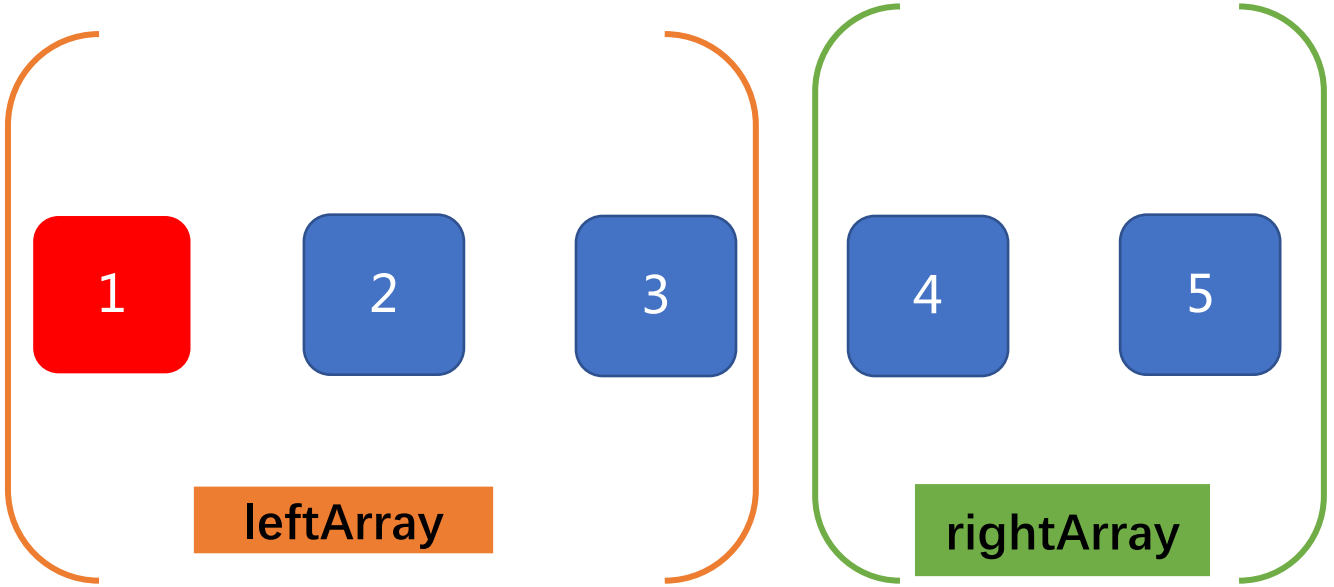
第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



将队列分为左右两个队列，左队列 leftArray 用黄括号括起来

LeetCode-1670 设计前中后队列

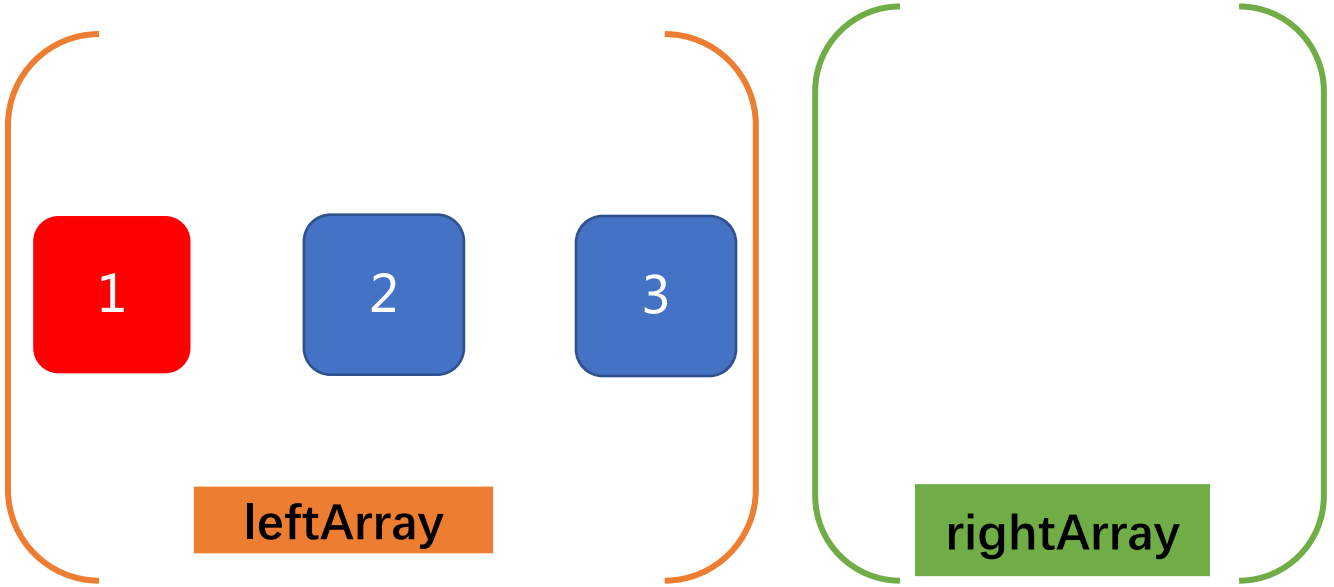
第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



右队列 rightArray 用绿括号括起来

LeetCode-1670 设计前中后队列

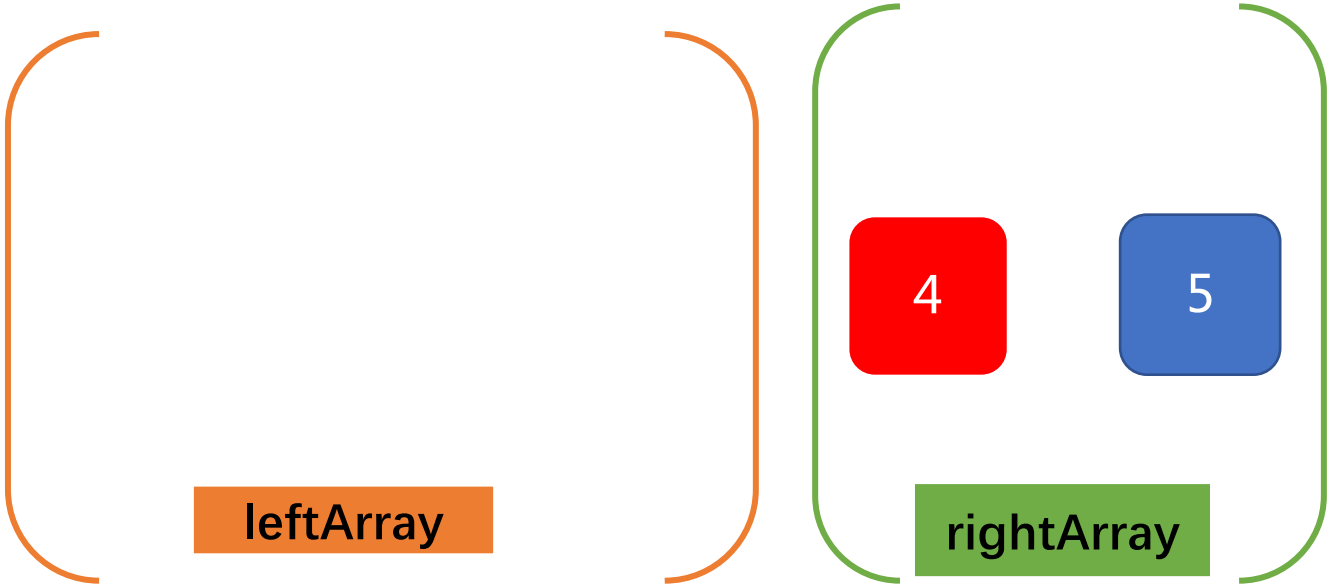
第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



先来判断一下：如果右队列长度不存在，返回 -1

LeetCode-1670 设计前中后队列

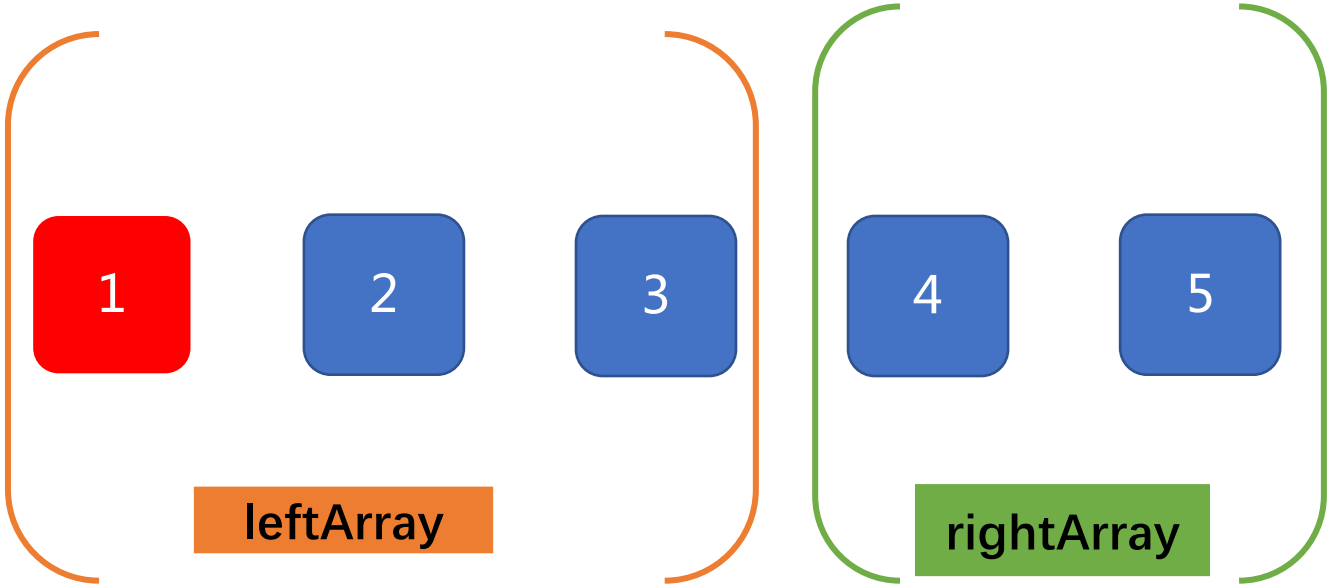
第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



同样也判断一下：如果左队列长度不存在，便是删除右队列的第一位，这里用到的方法是 shift

LeetCode-1670 设计前中后队列

第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1

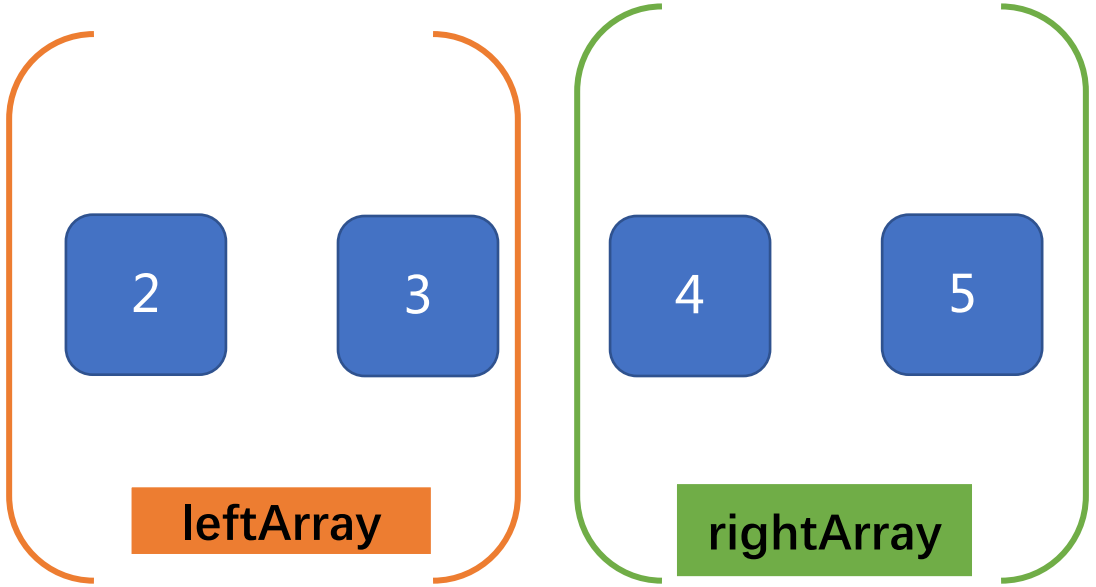


否则就是 左队列不为空，直接在左队列的第一位删除数据，用到了方法shift

shift **是用于删除第一位，并返回删除的数据**

LeetCode-1670 设计前中后队列

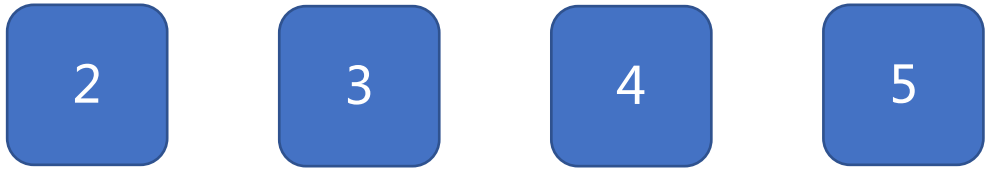
第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



所以，我们删除左队列的第一位

LeetCode-1670 设计前中后队列

第四个要求，将 最前面 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



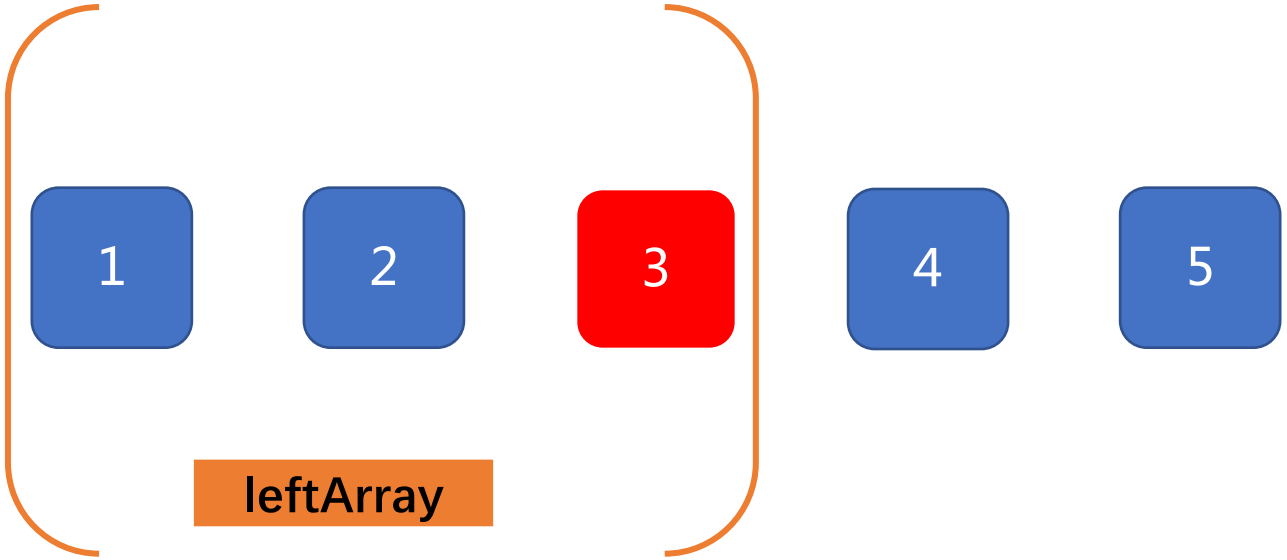
这便是删除队列的第一位的效果

LeetCode-1670 设计前中后队列

第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1

LeetCode-1670 设计前中后队列

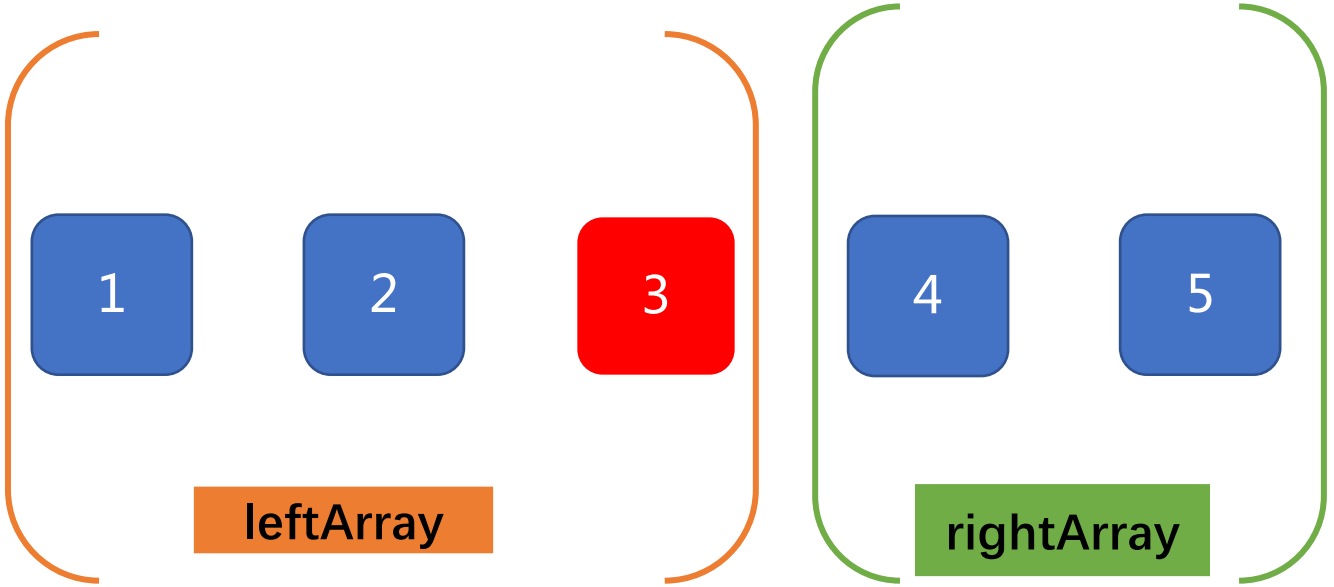
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



将队列分为左右两个队列，左队列 leftArray 用黄括号括起来

LeetCode-1670 设计前中后队列

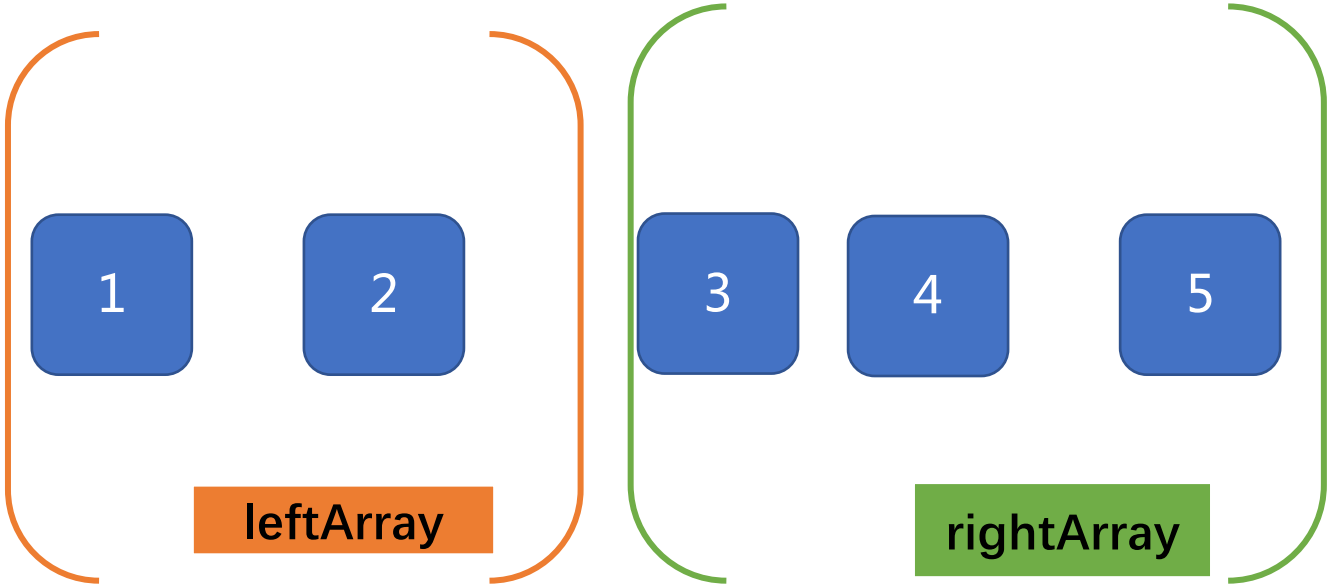
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



右队列 rightArray 用绿括号括起来

LeetCode-1670 设计前中后队列

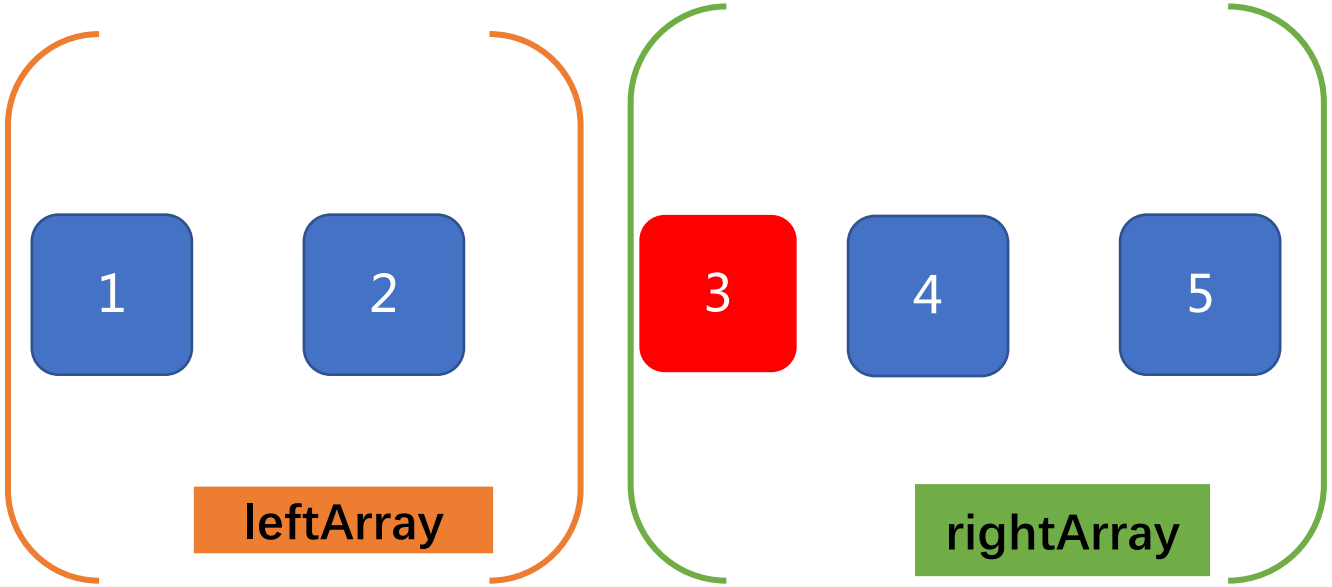
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



先来判断一下：如果左队列的长度 小于 右队列的长度 ，我们便删除，右队列的第一位，用到了 shift

LeetCode-1670 设计前中后队列

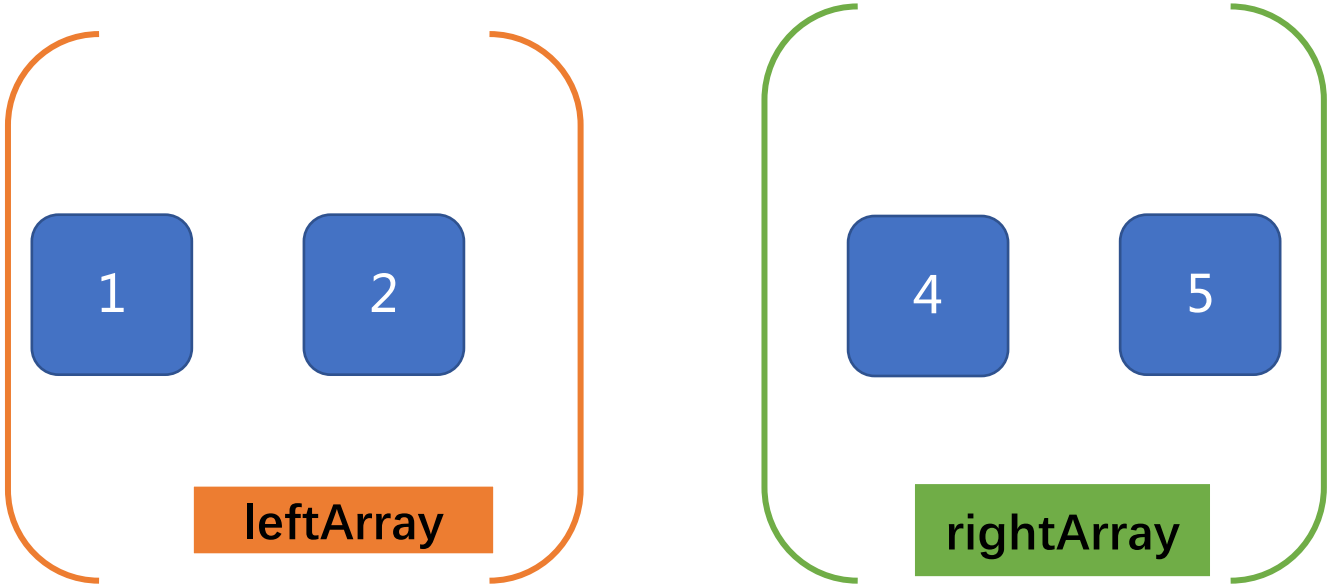
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



先来判断一下：如果左队列的长度 小于 右队列的长度 ，我们便删除，右队列的第一位，用到了 shift

LeetCode-1670 设计前中后队列

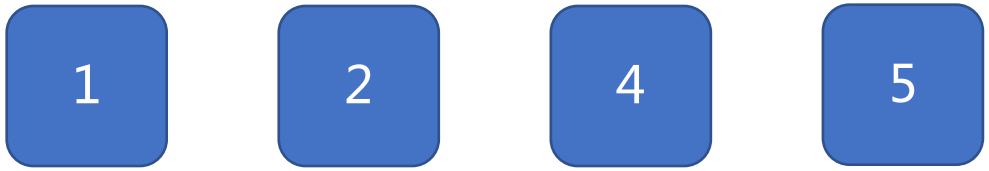
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



这里便删除了

LeetCode-1670 设计前中后队列

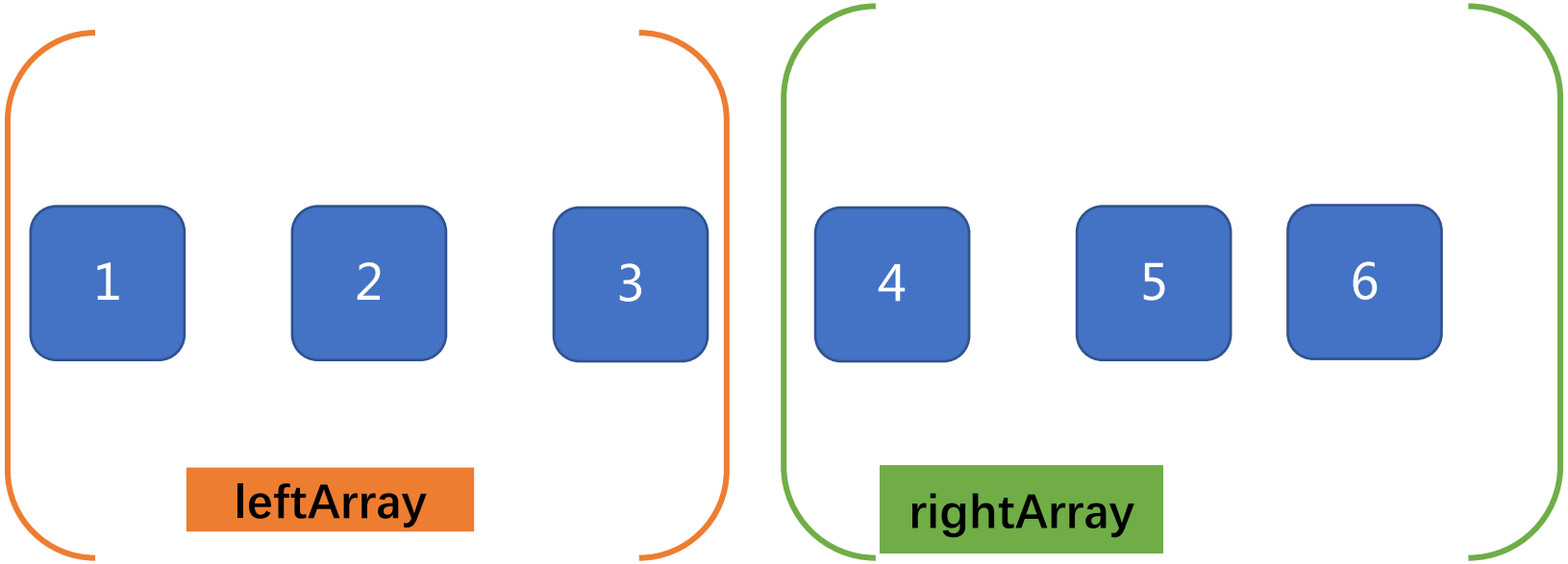
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



这里便是删除队列的正中间

LeetCode-1670 设计前中后队列

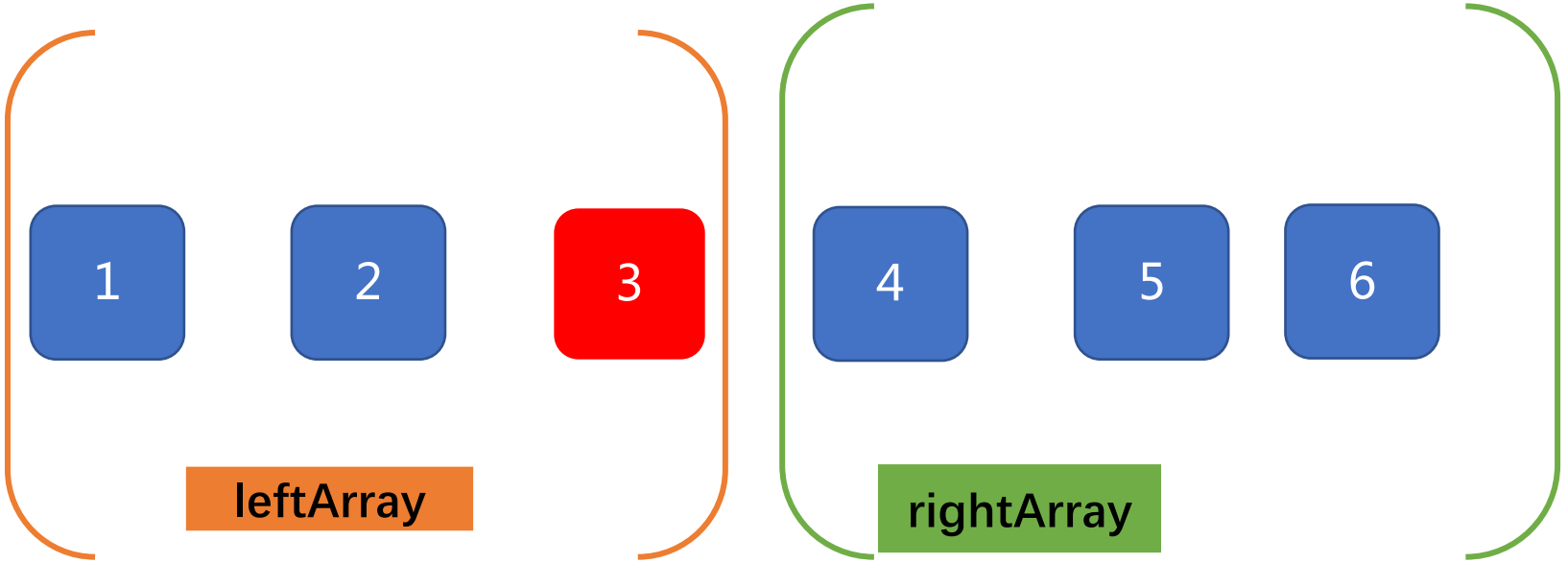
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



再来判断一下：如果左队列的长度 和 右队列的长度相等，我们便删除，左队列的最后一位，用到了pop

LeetCode-1670 设计前中后队列

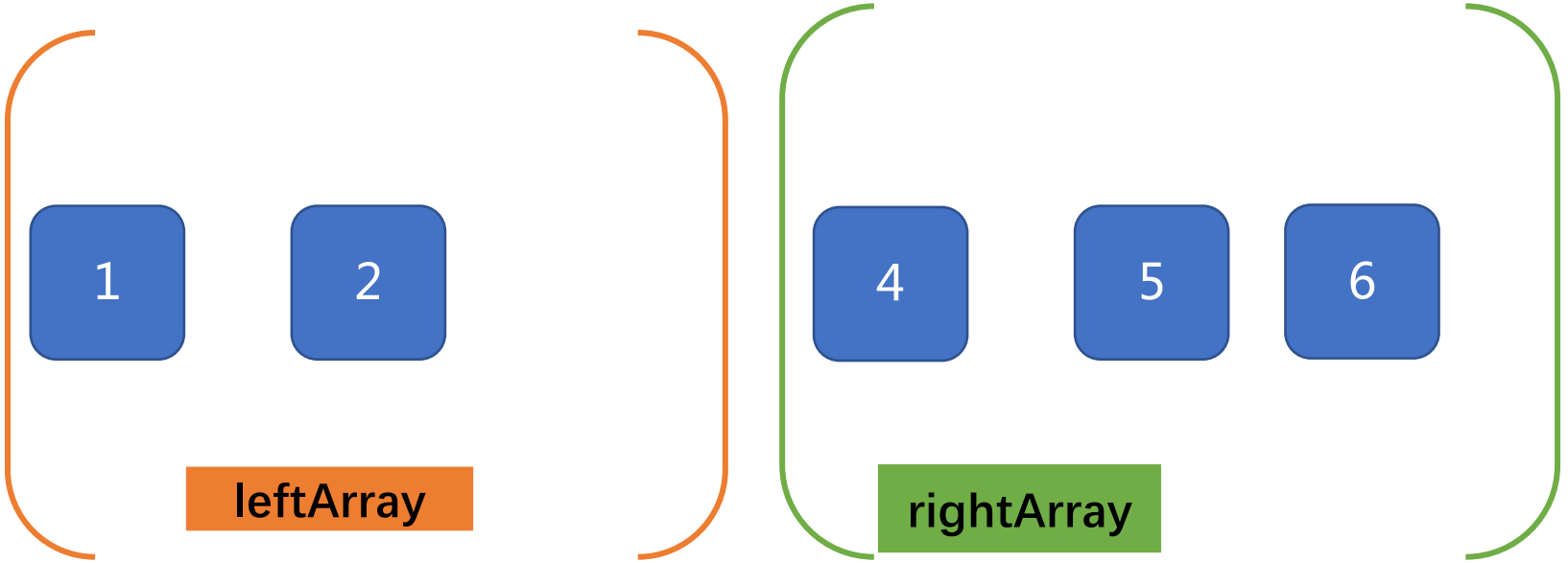
第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



再来判断一下：如果左队列的长度 和 右队列的长度相等，我们便删除，左队列的最后一位，用到了pop

LeetCode-1670 设计前中后队列

第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



再来判断一下：如果左队列的长度 和 右队列的长度相等，我们便删除，左队列的最后一位，用到了pop

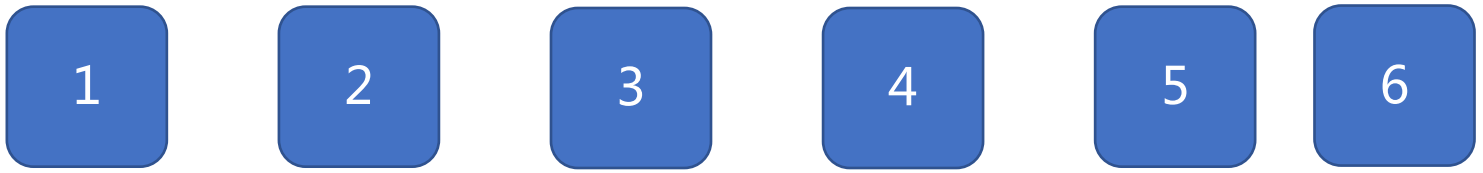
LeetCode-1670 设计前中后队列

第五个要求，将 正中间 的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1

这便是删除正中间最终效果

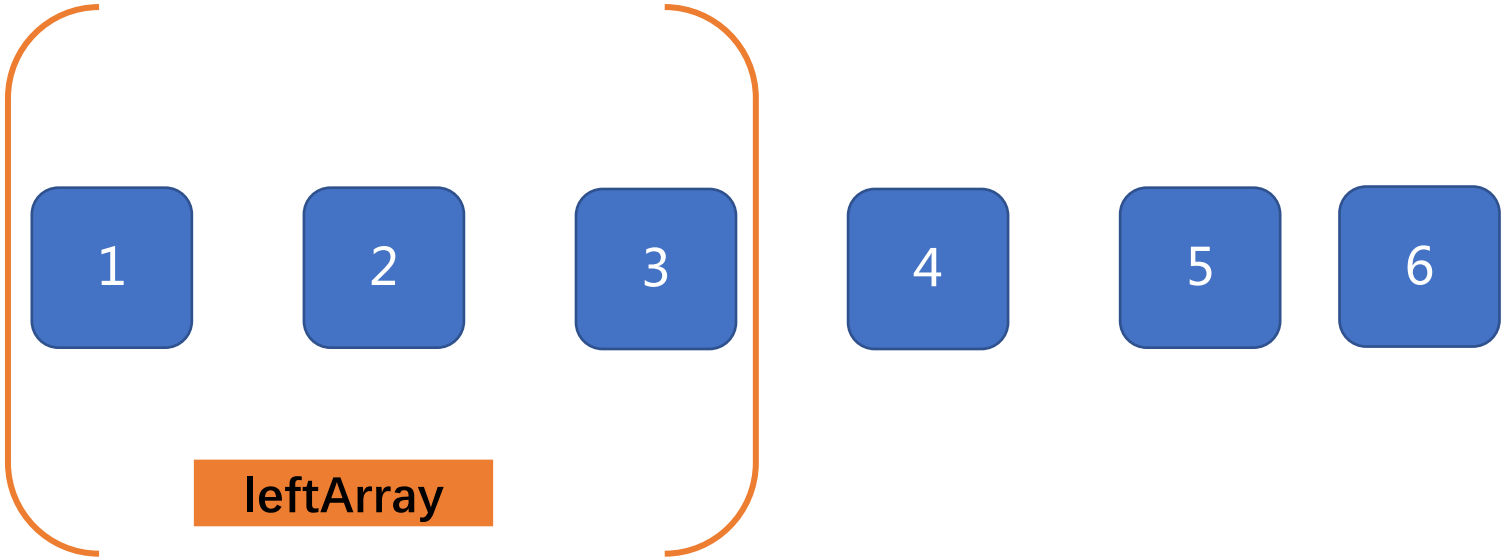
LeetCode-1670 设计前中后队列

第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



LeetCode-1670 设计前中后队列

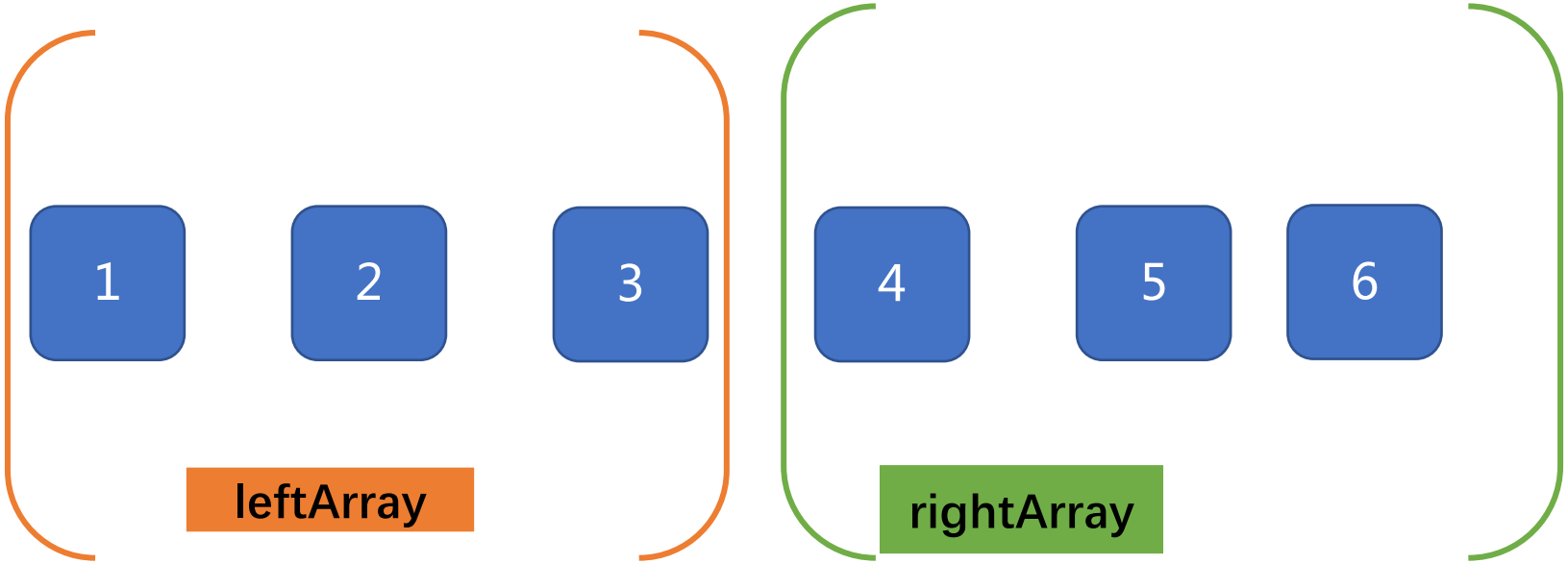
第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



将队列分为左右两个队列，左队列 leftArray 用黄括号括起来

LeetCode-1670 设计前中后队列

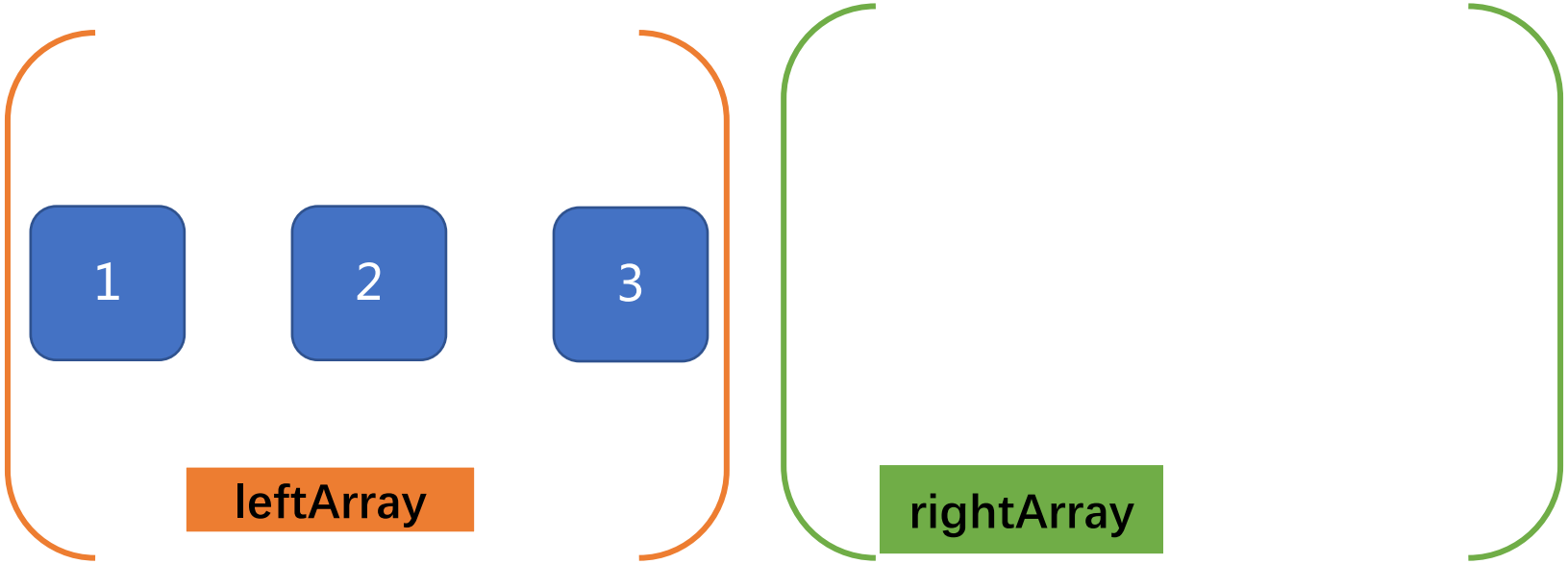
第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



右队列 rightArray 用绿括号括起来

LeetCode-1670 设计前中后队列

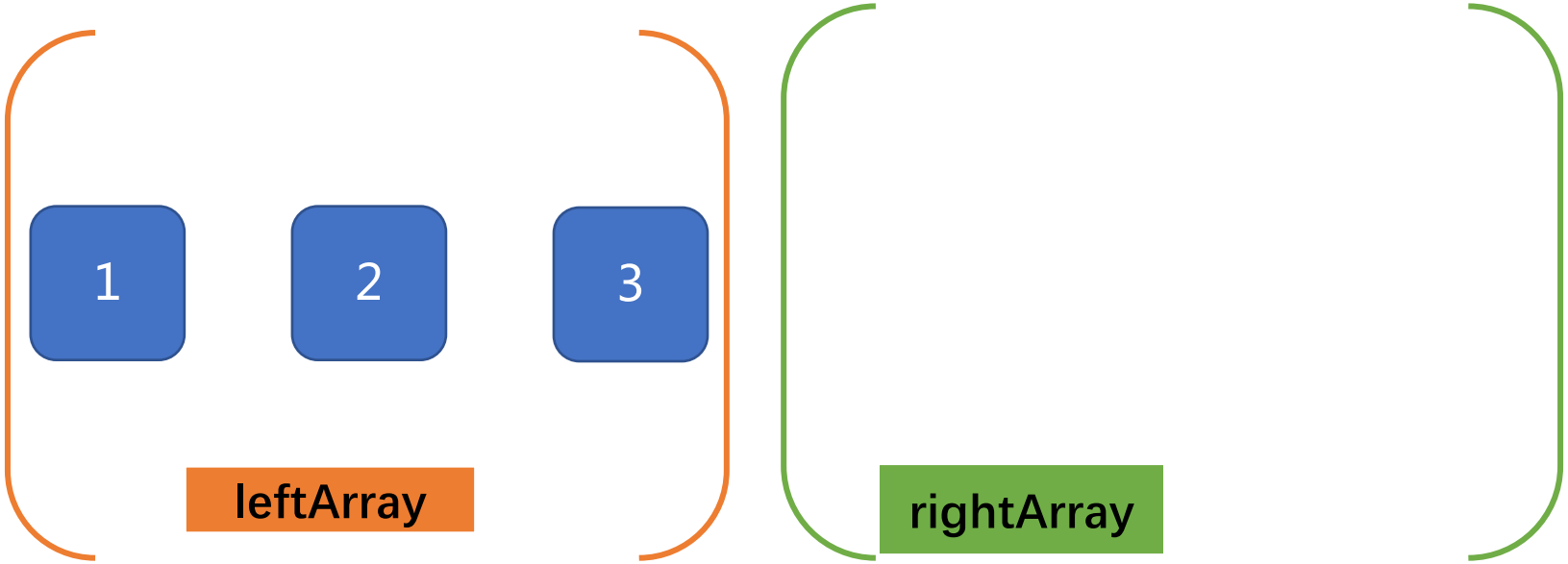
第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



接下来判断一下，如果右队列是空的

LeetCode-1670 设计前中后队列

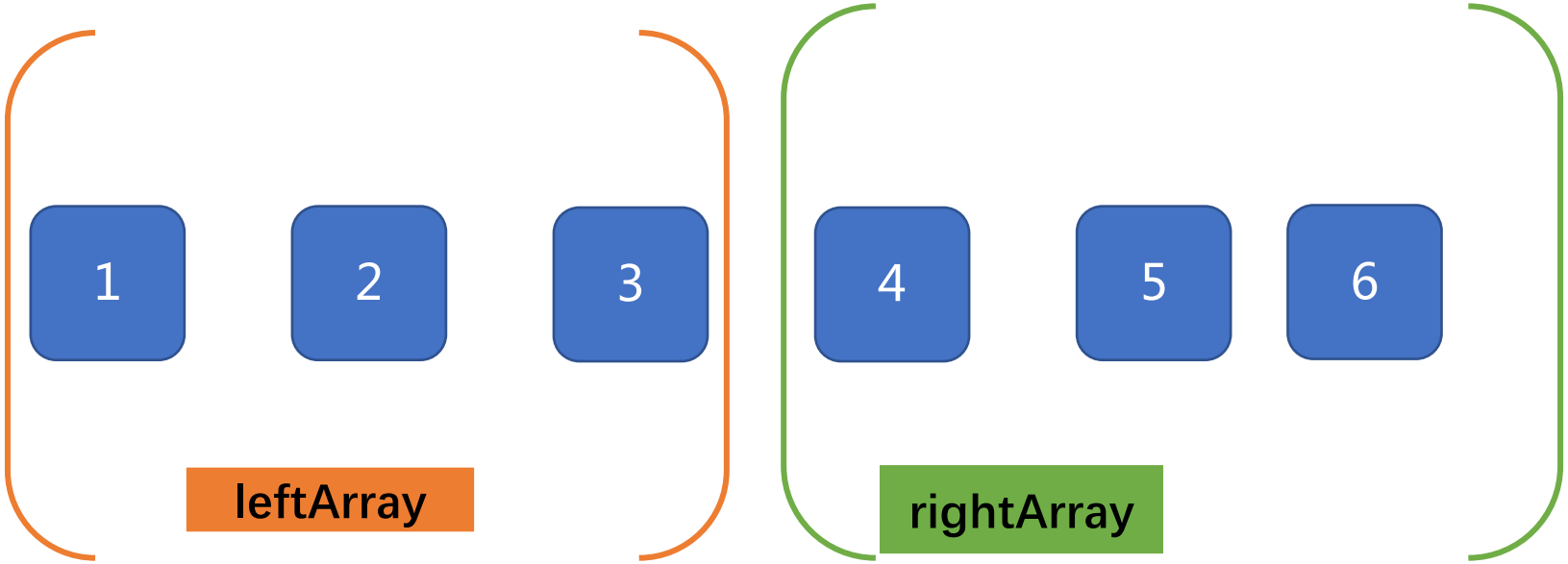
第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



返回 -1

LeetCode-1670 设计前中后队列

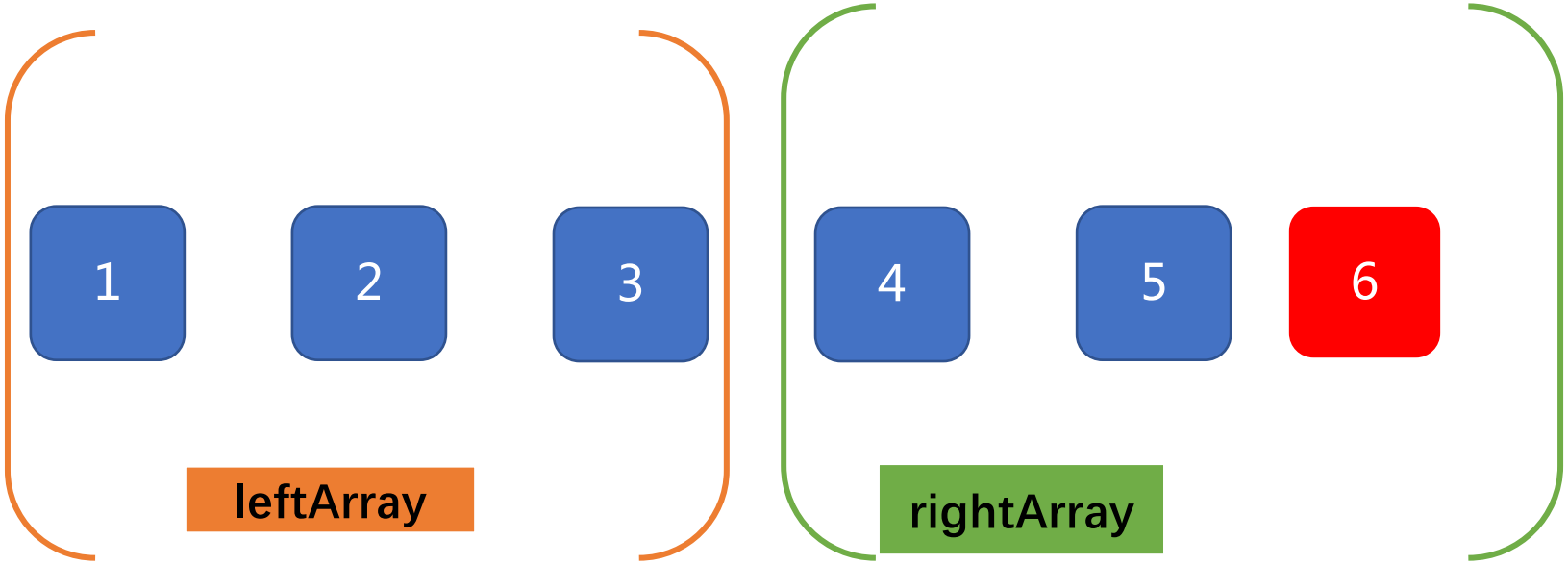
第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



否则，右队列不是空的，就让右队列删除最后一位，用到了方法 pop

LeetCode-1670 设计前中后队列

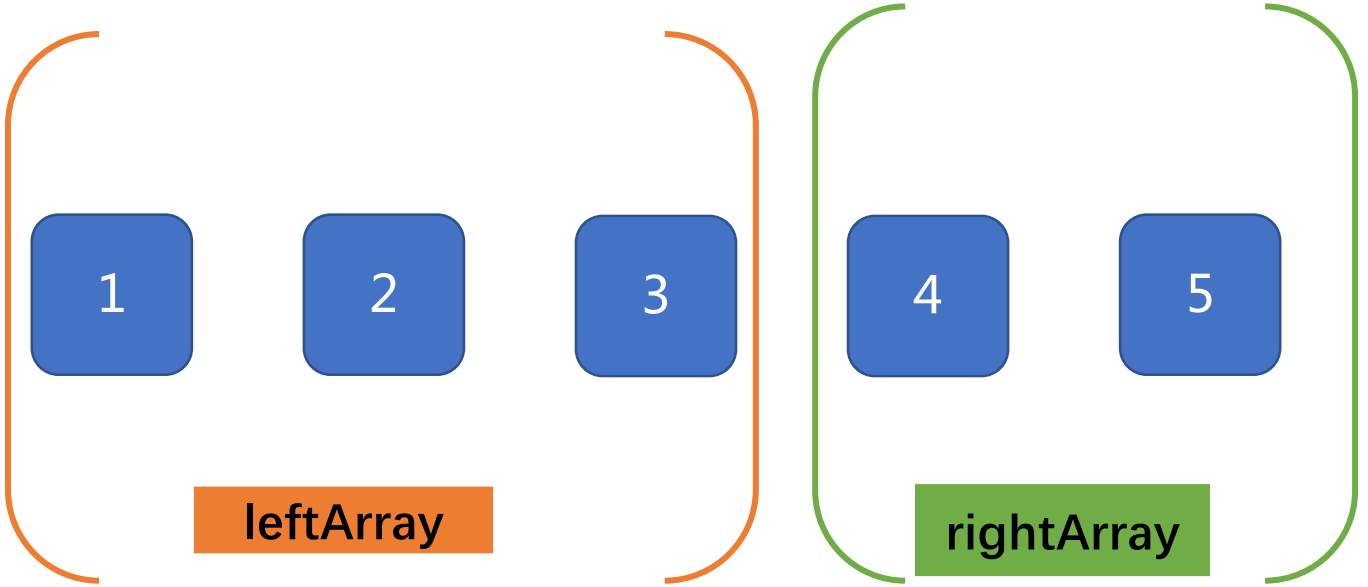
第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



否则，右队列不是空的，就让右队列删除最后一位，用到了方法 pop

LeetCode-1670 设计前中后队列

第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1



否则，右队列不是空的，就让右队列删除最后一位，用到了方法 pop

LeetCode-1670 设计前中后队列

第六个要求，将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回 -1

这便是删除最后一位的效果

933.最近请求次数

门徒计划，带你开启算法精进之路

LeetCode-933 最近请求次数

经典面试题-智力发散题

大约用时：（45 mins）

下一部分：答疑解惑-留作业

17.09.第 k 个数

门徒计划，带你开启算法精进之路

LeetCode-17.09 第 k 个数

859.亲密字符串

门徒计划，带你开启算法精进之路

LeetCode-859 亲密字符串

A = aaaaabc

B = aaaaacb

这是我们将要比较的两个字符串

LeetCode-859 亲密字符串

A = aaaaabc
B = aaaaacb

只有两种情况我们才把这两个字符串判定为亲密字符串
1.只有两处不同，并且两处不同是可交换的，如ab和ba
2.A和B完全相同，并且至少有一个字符出现次数为两次及以上

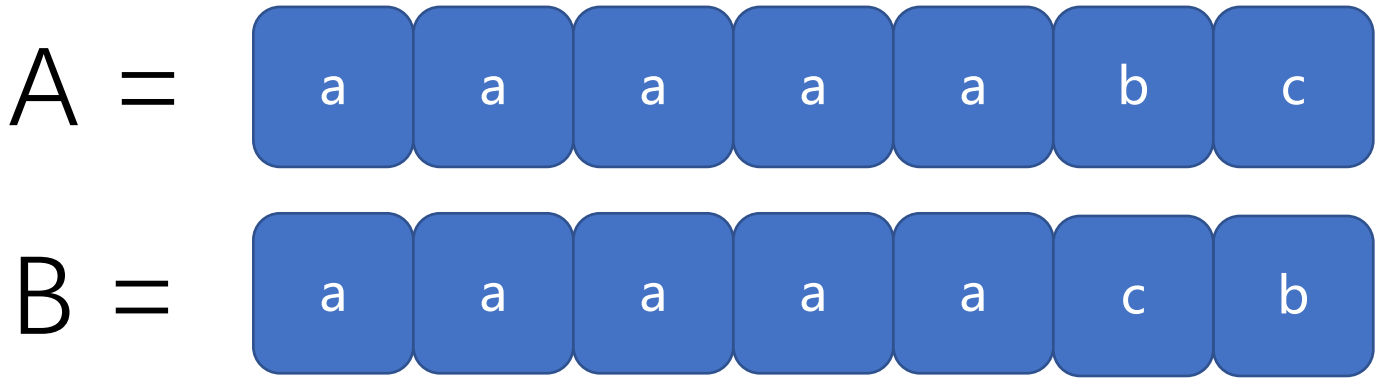
LeetCode-859 亲密字符串

A = aaaaabc
B = aaaaacb

然后开始比较A，B字符串的每个同位置的字符
如果字符不一样，就记录两个字符，然后为后续比较做准备。

LeetCode-859 亲密字符串

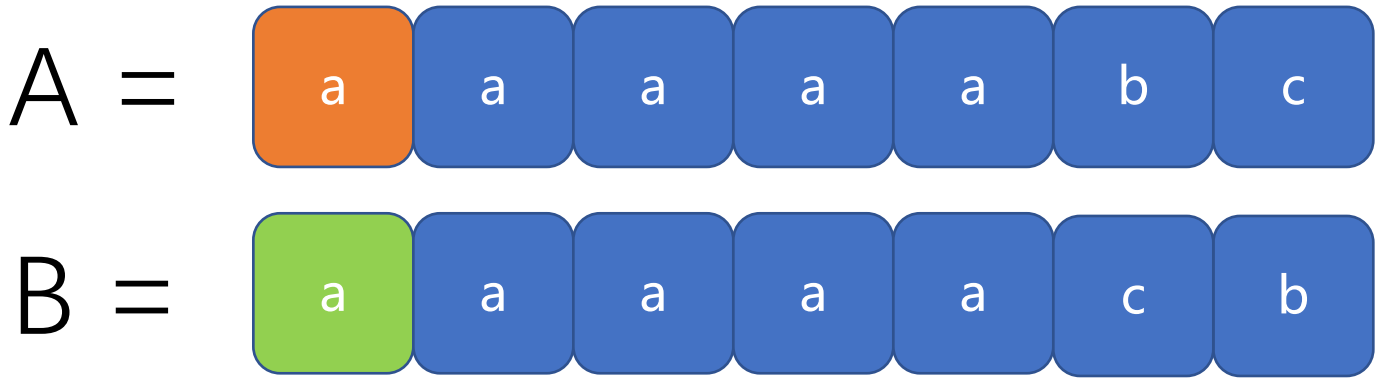
changeA = undefined　　flag = false
changeB = undefined　　duplicate[]= null



我们开始寻找A字符串和B字符串之间不同的位置

LeetCode-859 亲密字符串

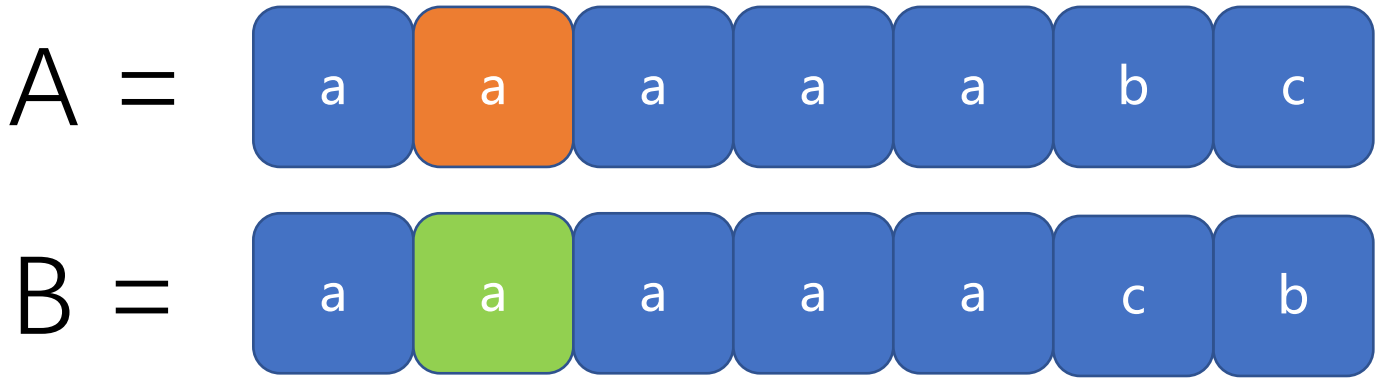
changeA = undefined　　flag = false
changeB = undefined　　duplicate[]= null



我们开始比较A,B字符串的字符

LeetCode-859 亲密字符串

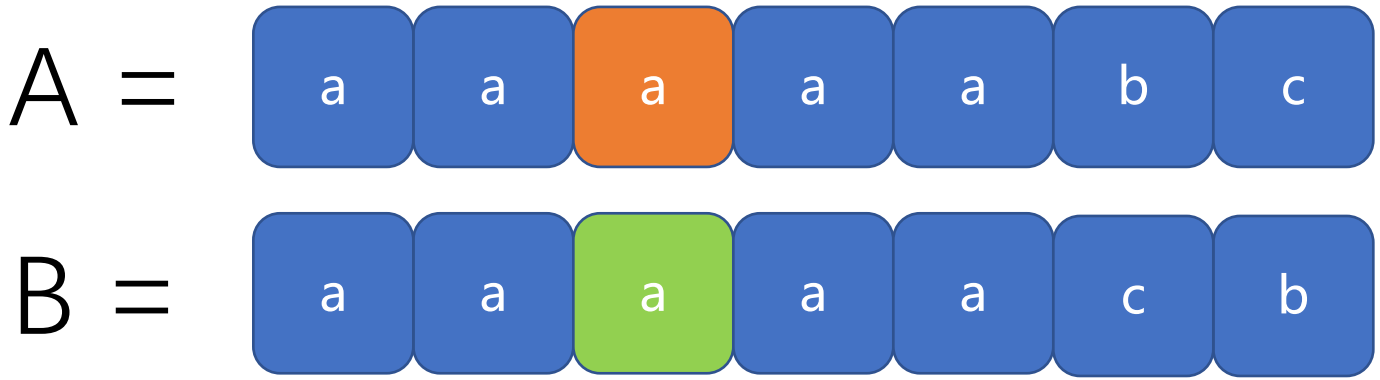
changeA = undefined　　flag = false
changeB = undefined　　duplicate[]= null



我们开始比较A,B字符串的字符

LeetCode-859 亲密字符串

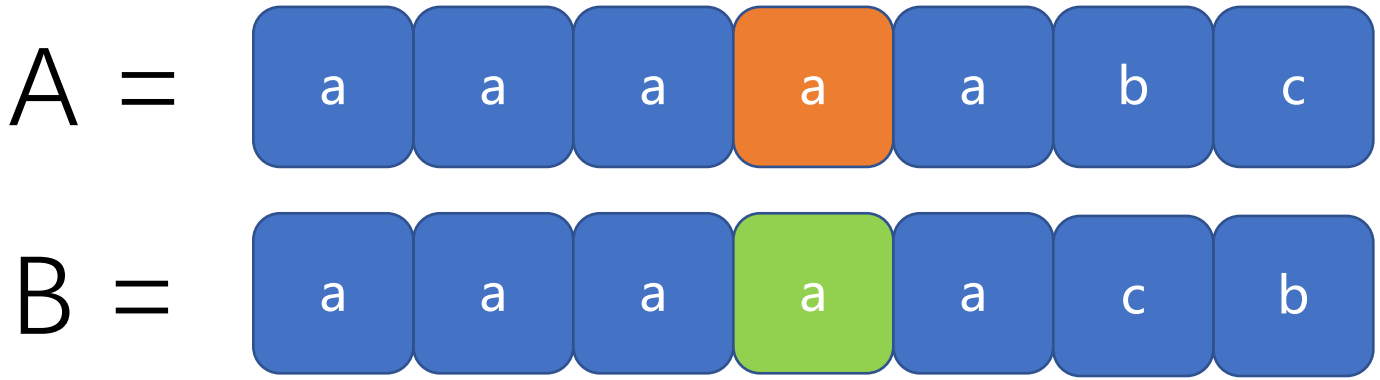
changeA = undefined　　flag = true
changeB = undefined　　duplicate[]= [a]



我们开始比较A,B字符串的字符

LeetCode-859 亲密字符串

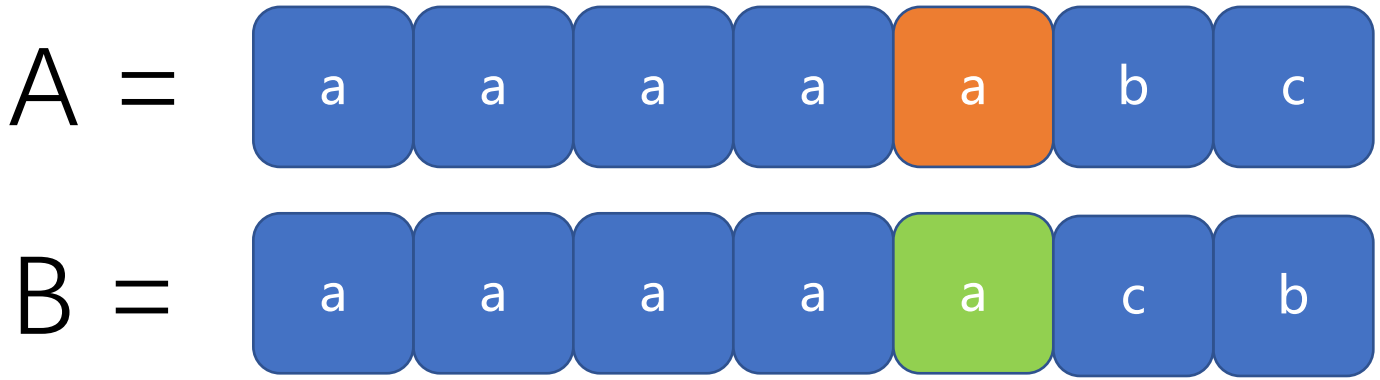
changeA = undefined　　flag = true
changeB = undefined　　duplicate[]= [a]



我们开始寻找A字符串和B字符串之间不同的位置

LeetCode-859 亲密字符串

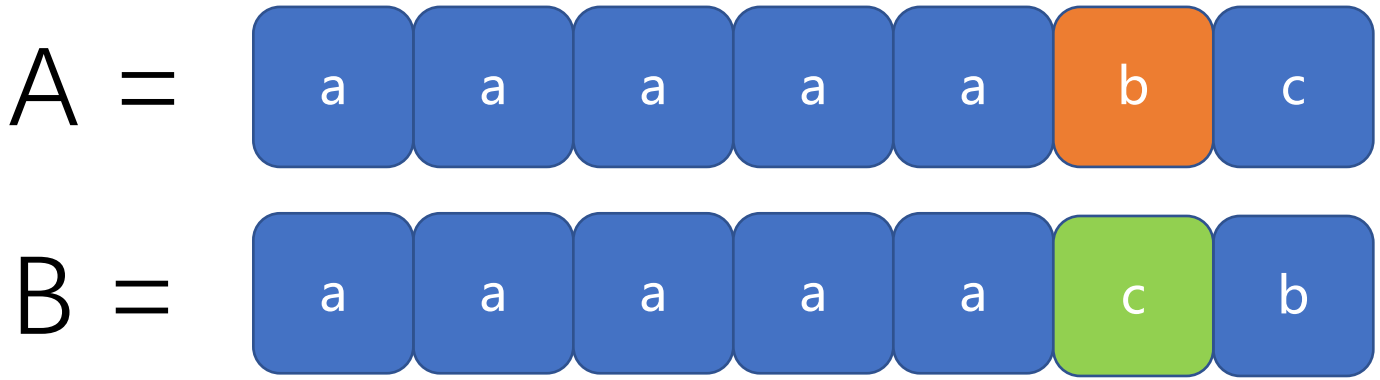
changeA = undefined　　flag = true
changeB = undefined　　duplicate[]= [a]



我们开始寻找A字符串和B字符串之间不同的位置

LeetCode-859 亲密字符串

changeA = undefined　　flag = true
changeB = undefined　　duplicate[]= [a]

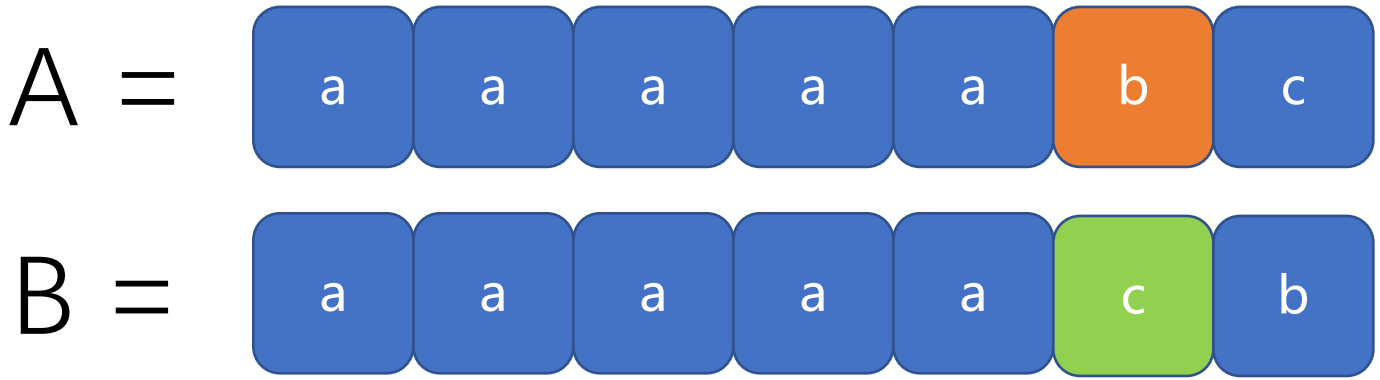


当我们找到第一处不同点的时候，
我们先判断下我们的changA是否为undefined，保证这是我们第一处不同点

LeetCode-859 亲密字符串

changeA = b
changeB = c

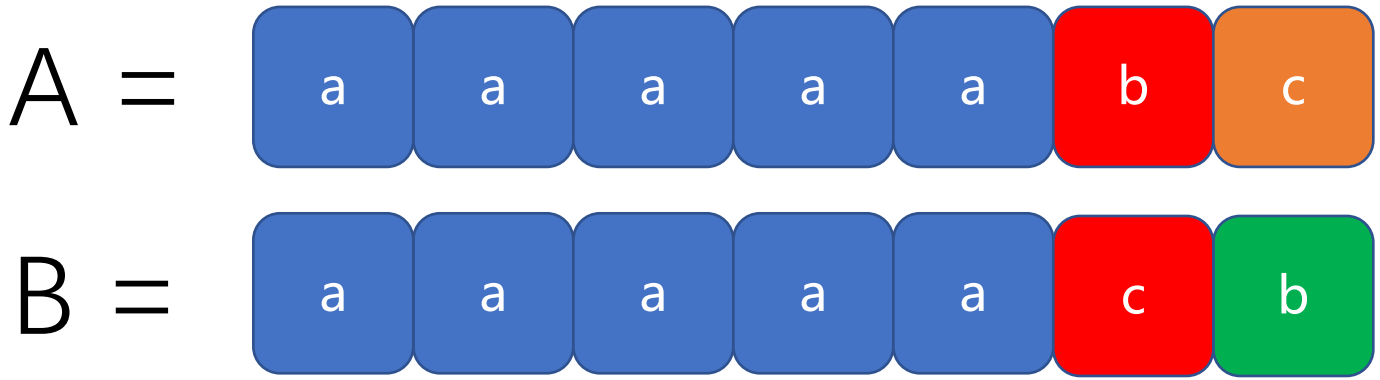
flag = true
duplicate[]= [a]



如果changeA为undefined证明我们目前为第一个不同点
就将当前的A和B的字符，赋值给changeA和changeB

changeA = b
changeB = c

flag = true
duplicate[]= [a]

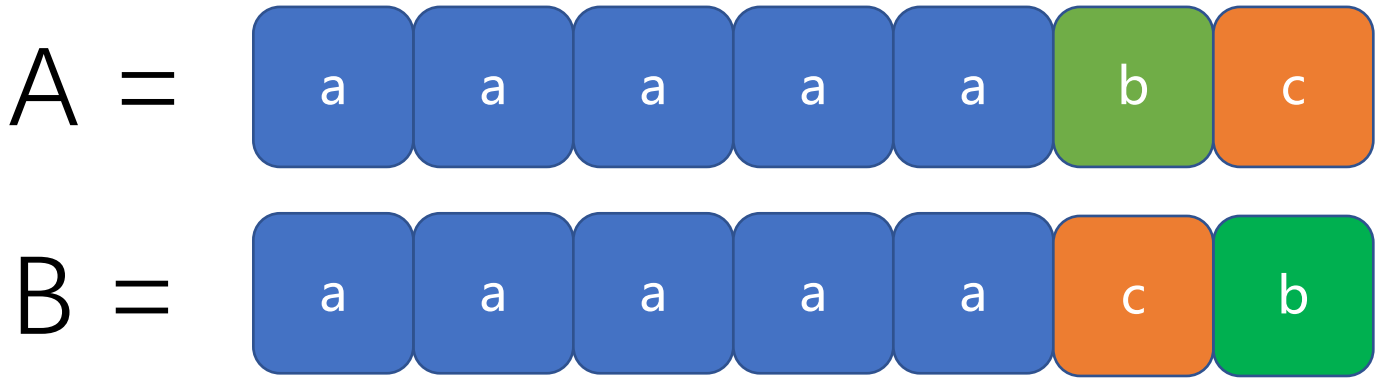


然后继续找下一处不同的位置

LeetCode-859 亲密字符串

changeA = b
changeB = c

flag = true
duplicate[]= [a]

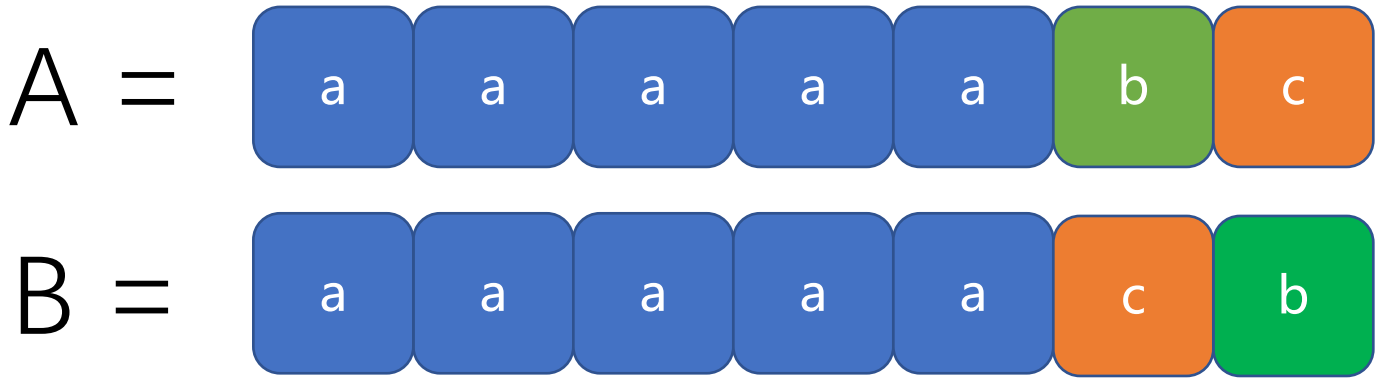


判断changeA是否和字符串B第二处不同的字符是否相等
changeB是否和字符串A第二处不同的字符是否相等

LeetCode-859 亲密字符串

changeA = b
changeB = c

flag = true
duplicate[]= [a]

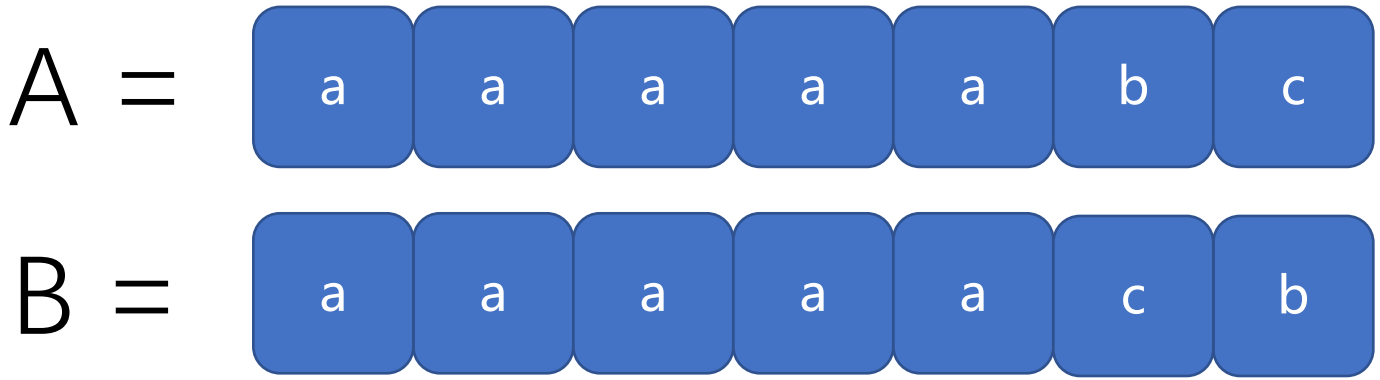


如果不相等就证明，这处不同点不能通过交换来保证字符串相等。
就违背了我们的第一条推理：只有两次不同，并且两次不同是可交换的，如ab和ba
它们就不是亲密字符串，就返回false

LeetCode-859 亲密字符串

changeA = null
changeB = null

flag = true
duplicate[]= [a]



当两个字符串的字符都比较完成，我们就判断changeA是否为null或者changeA为undefined并且flag为true

860.柠檬水找零

门徒计划，带你开启算法精进之路

LeetCode-17.09 第 k 个数

969.煎饼排序

门徒计划，带你开启算法精进之路

LeetCode-969 煎饼排序

煎饼排序简单点来说，就是每次只能反转数组中的第一个元素到第K个元素的子数组
然后我们每完成反转一次，就将k值记录下来。
直到这个数组变成有序递增的，就输出我们记录的所有的k的值。

LeetCode-969 煎饼排序

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

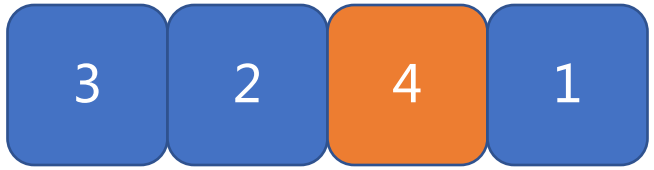
LeetCode-969 煎饼排序

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

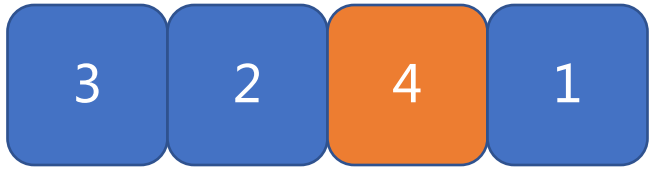
LeetCode-969 煎饼排序



我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,]



我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,2]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,2]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,2,3]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,2,3,]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,2,3,2]

我们首先找到数组中的最大值，先将其反转至数组头部，再将其反转到该元素排序后的位置；
然后找数组中的次大值，重复上述操作，直到数组排序完毕;

LeetCode-969 煎饼排序

K=[3,4,2,3,2]

整个数组完毕后，输出我们记录的K

621.任务调度

门徒计划，带你开启算法精进之路

LeetCode-621 任务调度

答疑解惑-留作业

大约用时：（5 mins）

下一部分：大家晚安

每天都想干翻这个世界
到头来，被世界干的服服帖帖

大家晚安
--船长