顺序表和链表

胡船长

初航我带你，远航靠自己

本章习题

1-应试. Leetcode-206：反转链表

2-应试. Leetcode-141：环形链表

3-校招.Leetcode-202：快乐数

4-校招.Leetcode-61：旋转链表

5-校招.Leetcode-19：删除链表的倒数第 N 个节点

6-校招.Leetcode-142：环形链表 II

7-校招.Leetcode-92：反转链表 II

回顾：数据结构

```c
typedef struct Vector {
    int size, length;
    int *data;
} Vector;

void init(Vector *vec, int size) {
    vec->data = (int *)malloc(size * sizeof(int));
    vec->size = size;
    vec->length = 0;
}

void expand(Vector *vec) {
    vec->size *= 2;
    vec->data = (int *)realloc(vec->data, vec->size * sizeof(int));
    printf("expand\n");
}

int insert(Vector *vec, int loc, int value) {
    if (loc < 0 || loc > vec->length) {
        return ERROR;
    }
    if (vec->length >= vec->size) {
        expand(vec);
    }
    for (int i = vec->length; i > loc; --i) {
        vec->data[i] = vec->data[i - 1];
    }
    vec->data[loc] = value;
    ++vec->length;
    return OK;
}
```

数据结构 = ？

回顾：数据结构

```c
typedef struct Vector {
    int size, length;
    int *data;
} Vector;

void init(Vector *vec, int size) {
    vec->data = (int *)malloc(size * sizeof(int));
    vec->size = size;
    vec->length = 0;
}

void expand(Vector *vec) {
    vec->size *= 2;
    vec->data = (int *)realloc(vec->data, vec->size * sizeof(int));
    printf("expand\n");
}

int insert(Vector *vec, int loc, int value) {
    if (loc < 0 || loc > vec->length) {
        return ERROR;
    }
    if (vec->length >= vec->size) {
        expand(vec);
    }
    for (int i = vec->length; i > loc; --i) {
        vec->data[i] = vec->data[i - 1];
    }
    vec->data[loc] = value;
    ++vec->length;
    return OK;
}
```

数据结构 = 结构定义 + 结构操作

本期内容

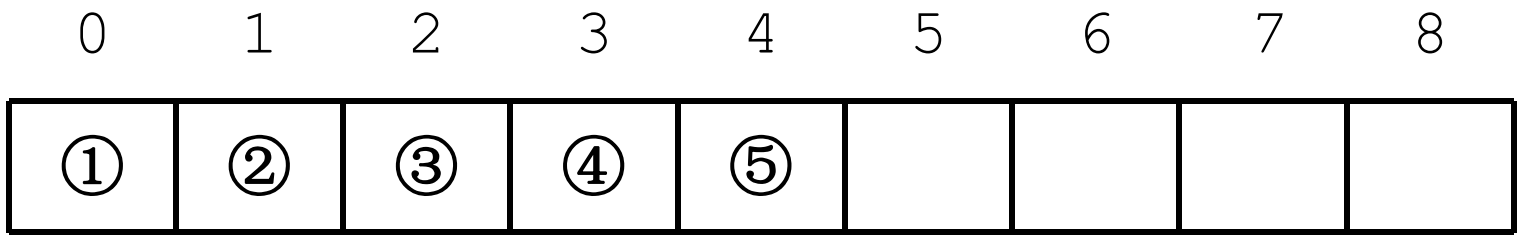
一. 顺序表：结构讲解 & 代码演示

二. 链表：结构讲解 & 代码演示

三. 循环链表 和 双向链表

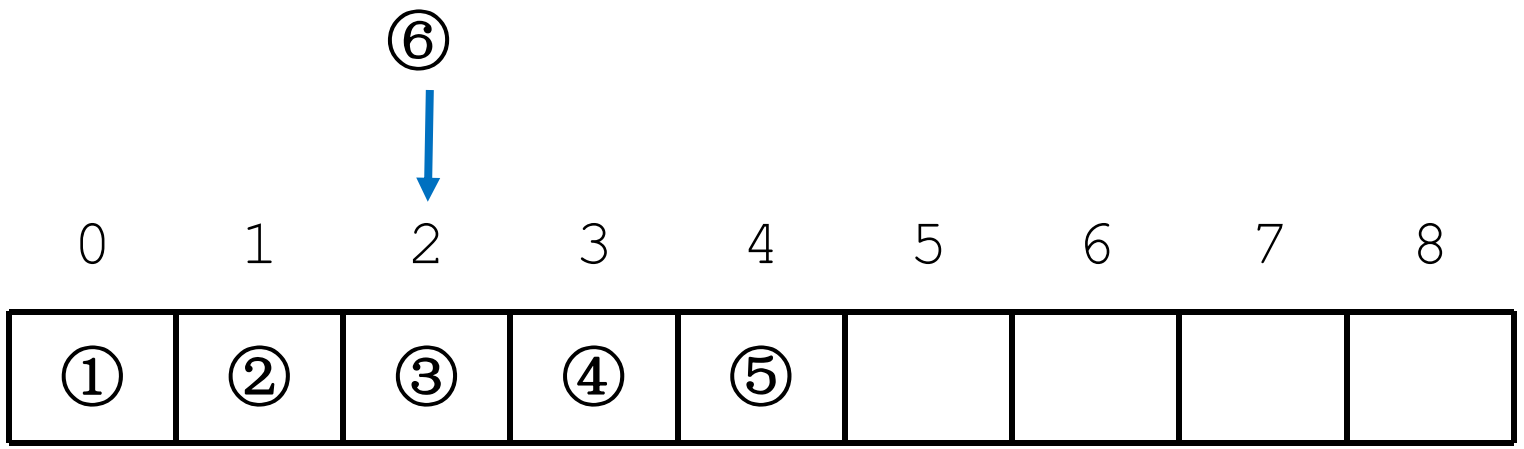
一. 顺序表：结构讲解 & 代码演示

顺序表：结构定义



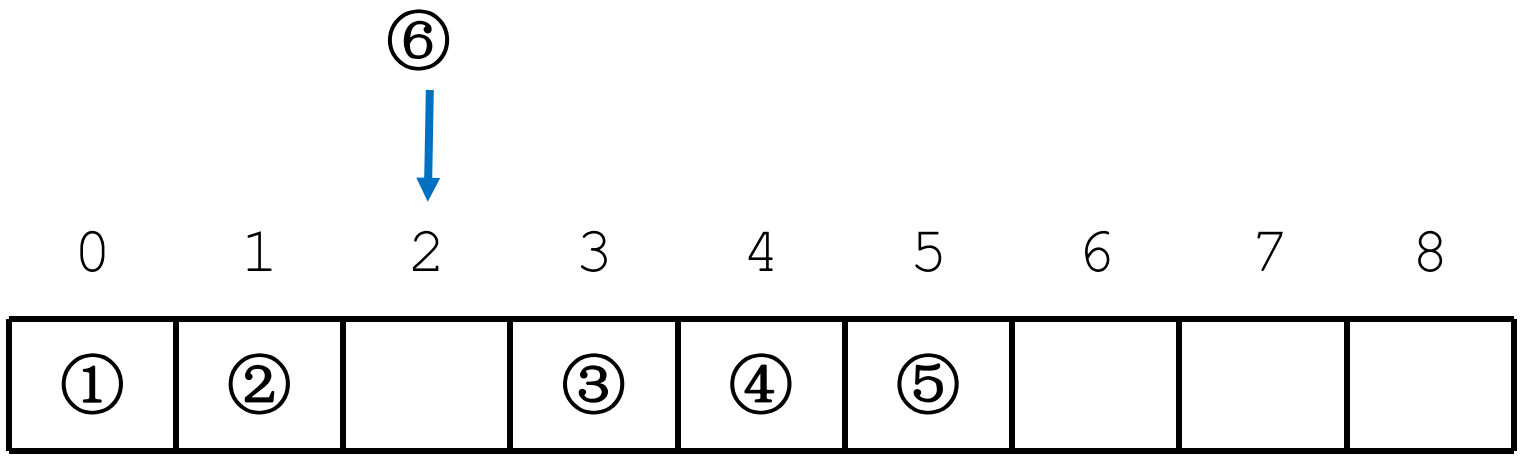
1、size = 9
2、count = 5

顺序表：插入演示



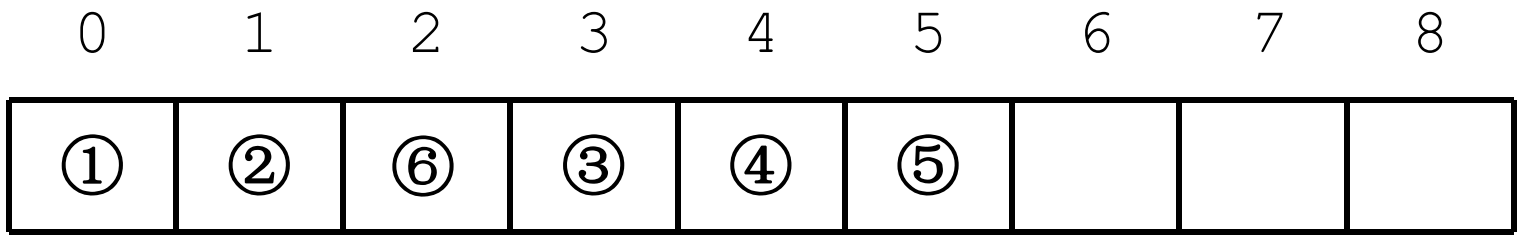
1、size = 9
2、count = 5

顺序表：插入演示



1、size = 9
2、count = 5

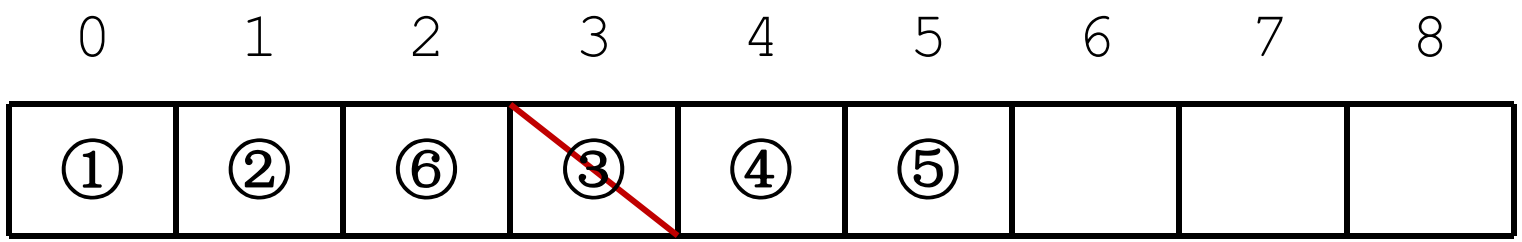
顺序表：插入演示



1、size = 9
2、count = 5

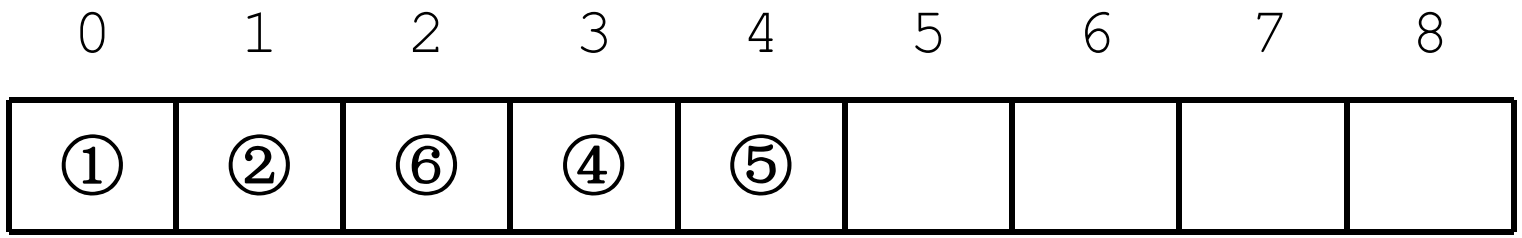
1、size = 9
2、count = 6

顺序表：删除演示



1、size = 9
2、count = 6

顺序表：删除演示



1、size = 9
2、count = 6

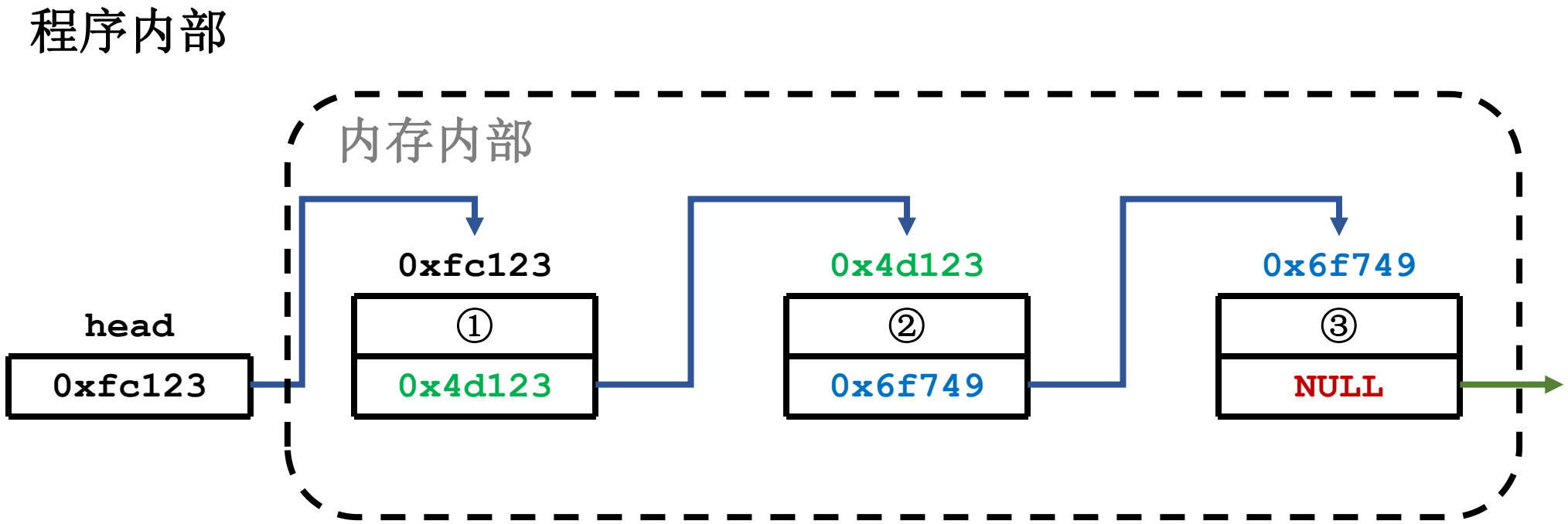
1、size = 9
2、count = 5


1. vim
vim ⌘1
bash ⌘2
bash ⌘3
39 }
40
41 Node *insert_maintain(Node *root) {
42 if (!hasRedChild(root)) return root;
43 if (root->lchild->color == RED && root->rchild->color == RED) {
44 if (!hasRedChild(root->lchild) && !hasRedChild(root->rchild)) return root;
45 root->color = RED;
46 root->lchild->color = root->rchild->color = BLACK;
47 return root;
48 }
49 if (root->lchild->color == RED) {
50 if (!hasRedChild(root->lchild)) return root;
51
52
53 } else {
54 if (!hasRedChild(root->rchild)) return root;
55
56 }
57
58 }
59
60
61 Node *__insert(Node *root, int key) {
62 if (root == NIL) return getNewNode(key);
<-6班 资 料 /X.现 场 撸 代 码 /15.RBT.cpp [FORMAT=unix] [TYPE=CPP] [POS=54,30][62%] 21/09/19 - 20:21

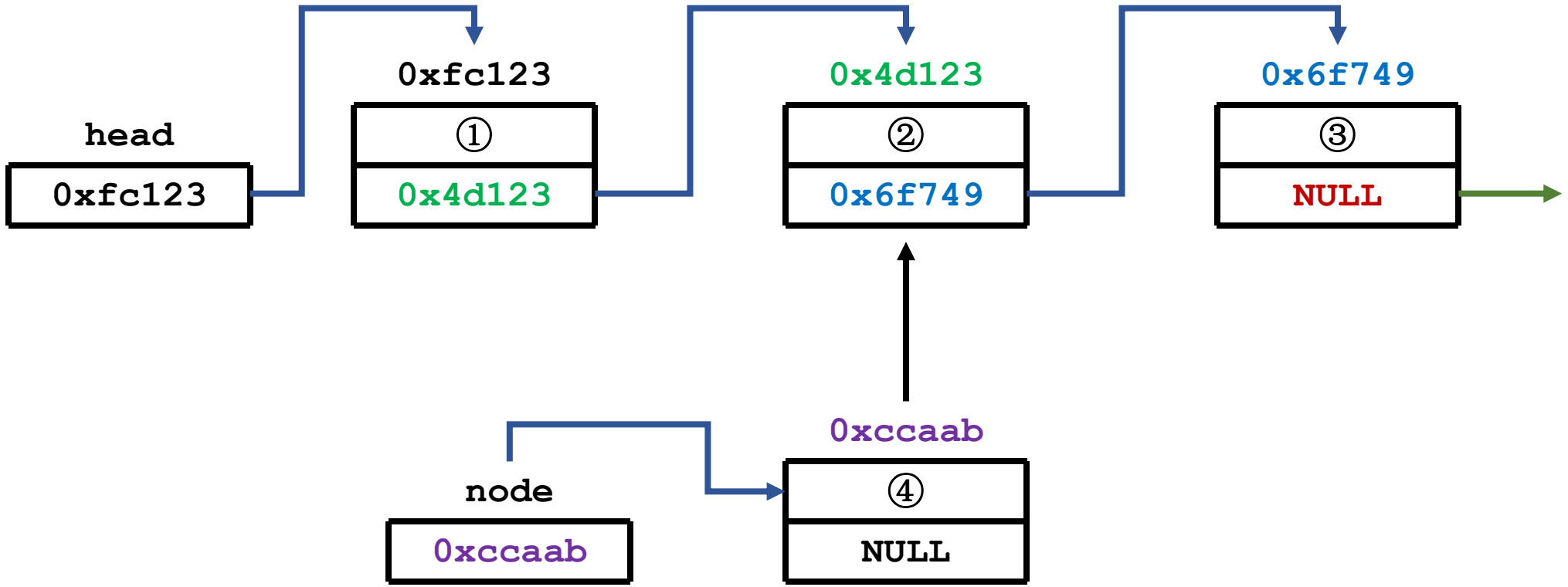
顺序表：代码演示

二. 链表：结构讲解 & 代码演示

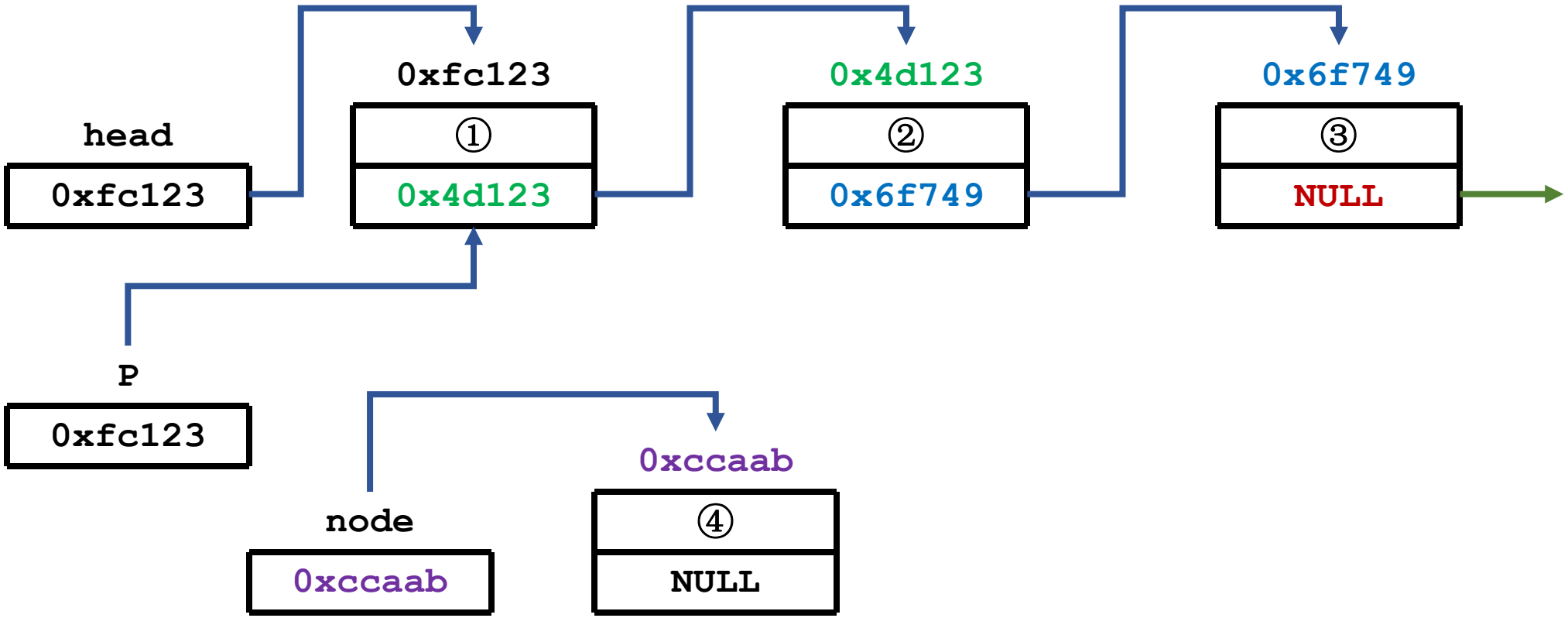
链表：结构定义


程序内部
内存内部
head
0xfc123
0xfc123
①
0x4d123
0x4d123
②
0x6f749
0x6f749
③
NULL

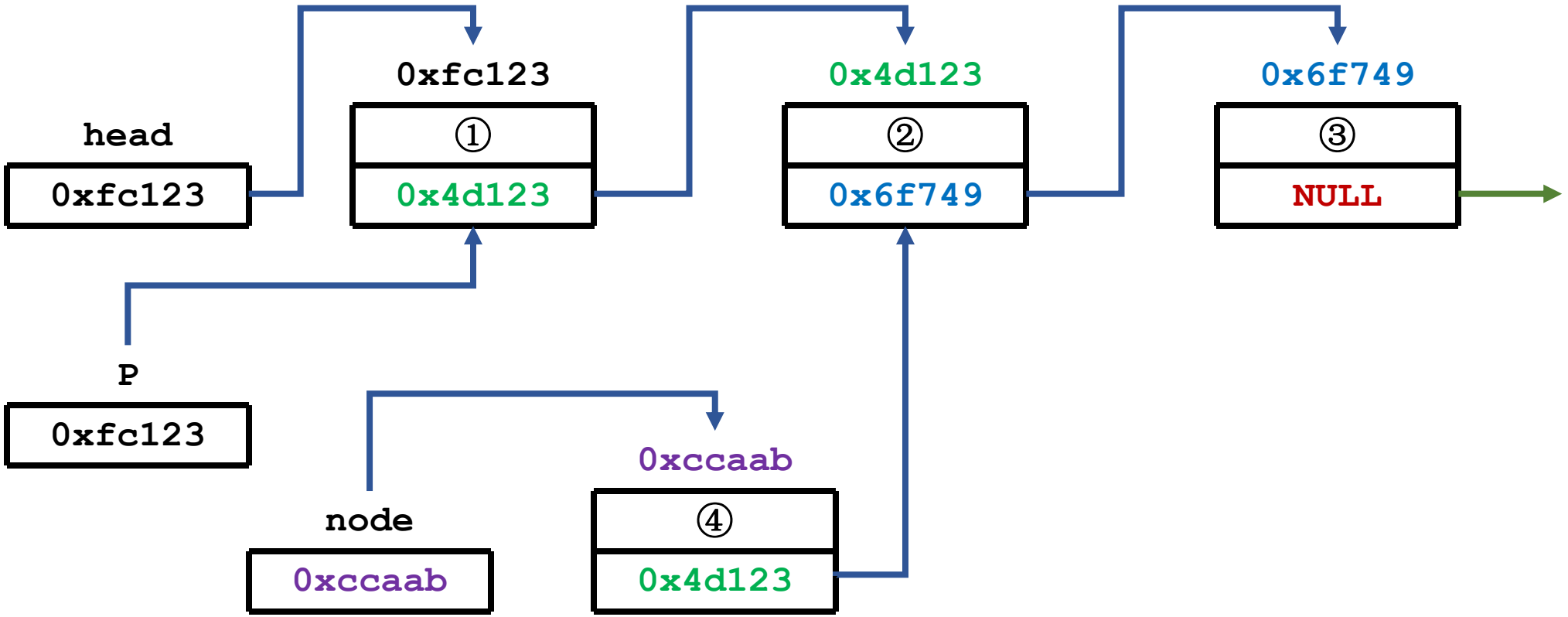
链表：插入元素


head
0xfc123
0xfc123
①
0x4d123
0x4d123
②
0x6f749
0x6f749
③
NULL
node
0xccaab
0xccaab
④
NULL

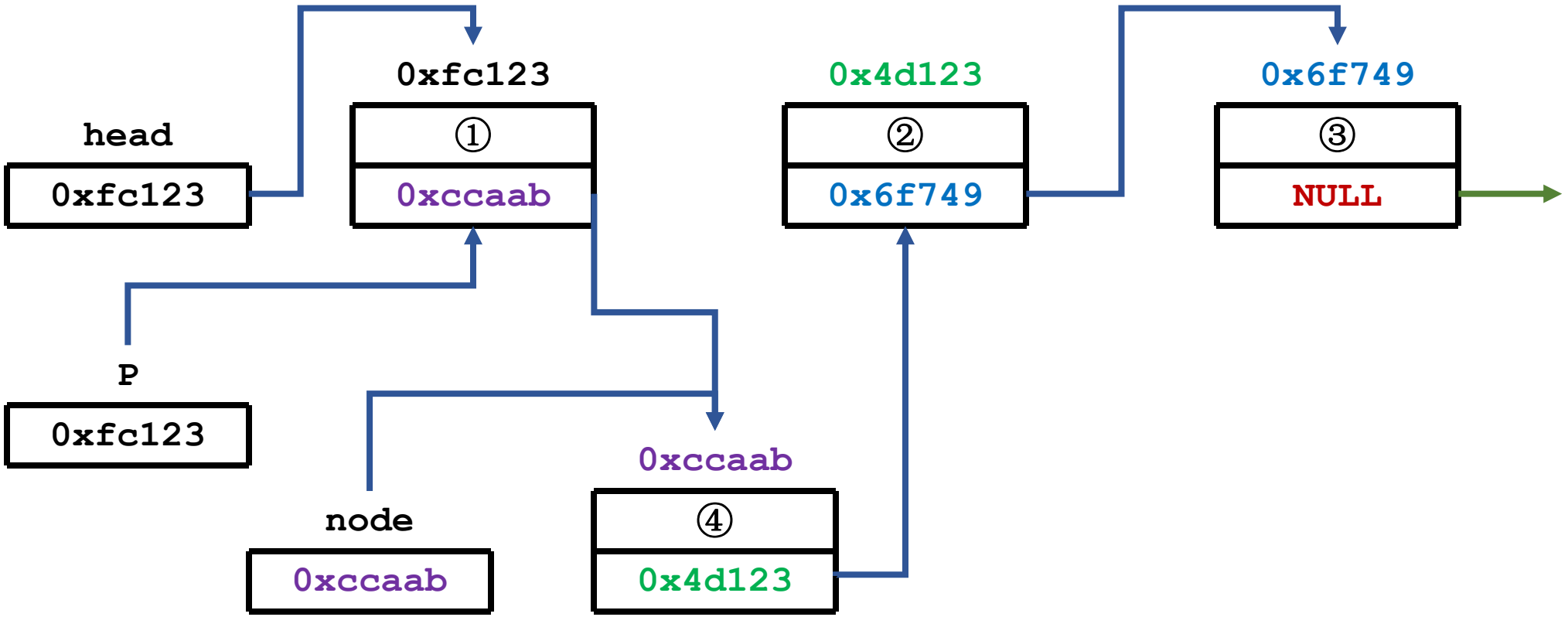
链表：插入元素


0xfc123
0x4d123
0x6f749
head
0xfc123
①
0x4d123
②
0x6f749
③
NULL
P
0xfc123
0xccaab
node
0xccaab
④
NULL

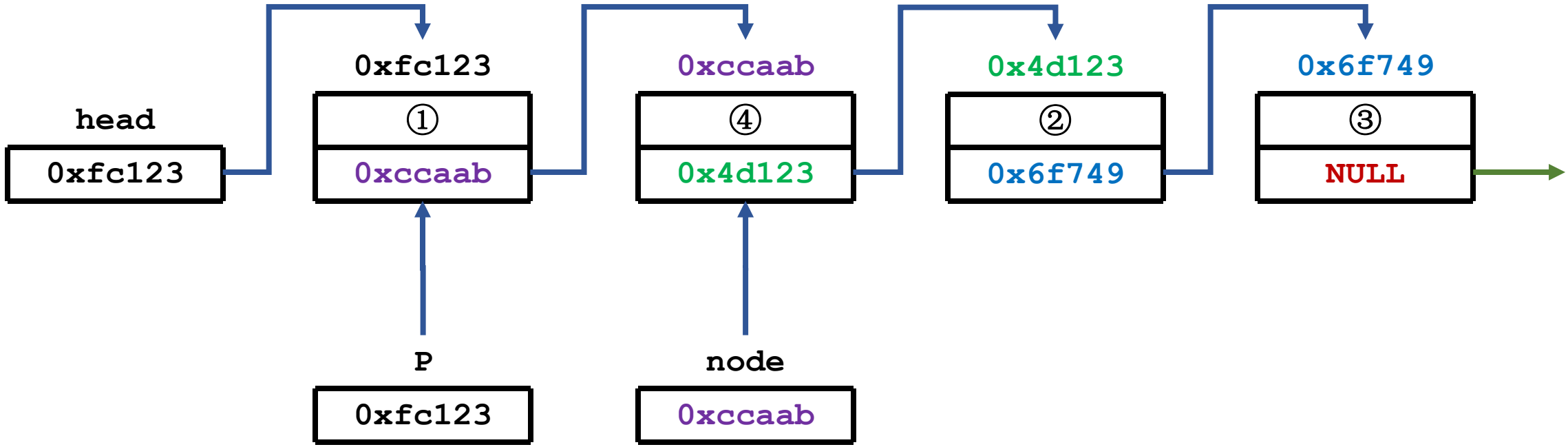
链表：插入元素


0xfc123
0x4d123
0x6f749
head
0xfc123
①
0x4d123
②
0x6f749
③
NULL
P
0xfc123
0xccaab
node
0xccaab
④
0x4d123

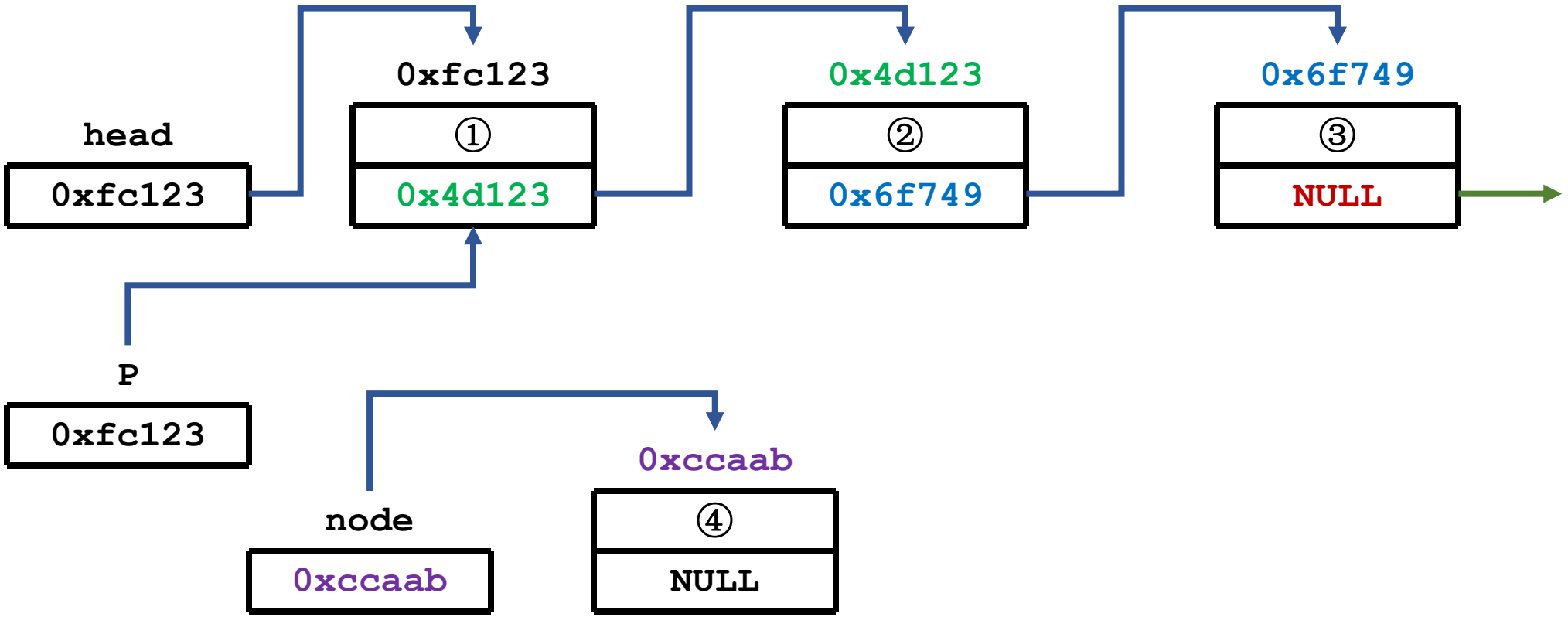
链表：插入元素


0xfc123
head
0xfc123
①
0xccaab
0x4d123
②
0x6f749
0x6f749
③
NULL
P
0xfc123
0xccaab
node
0xccaab
④
0x4d123

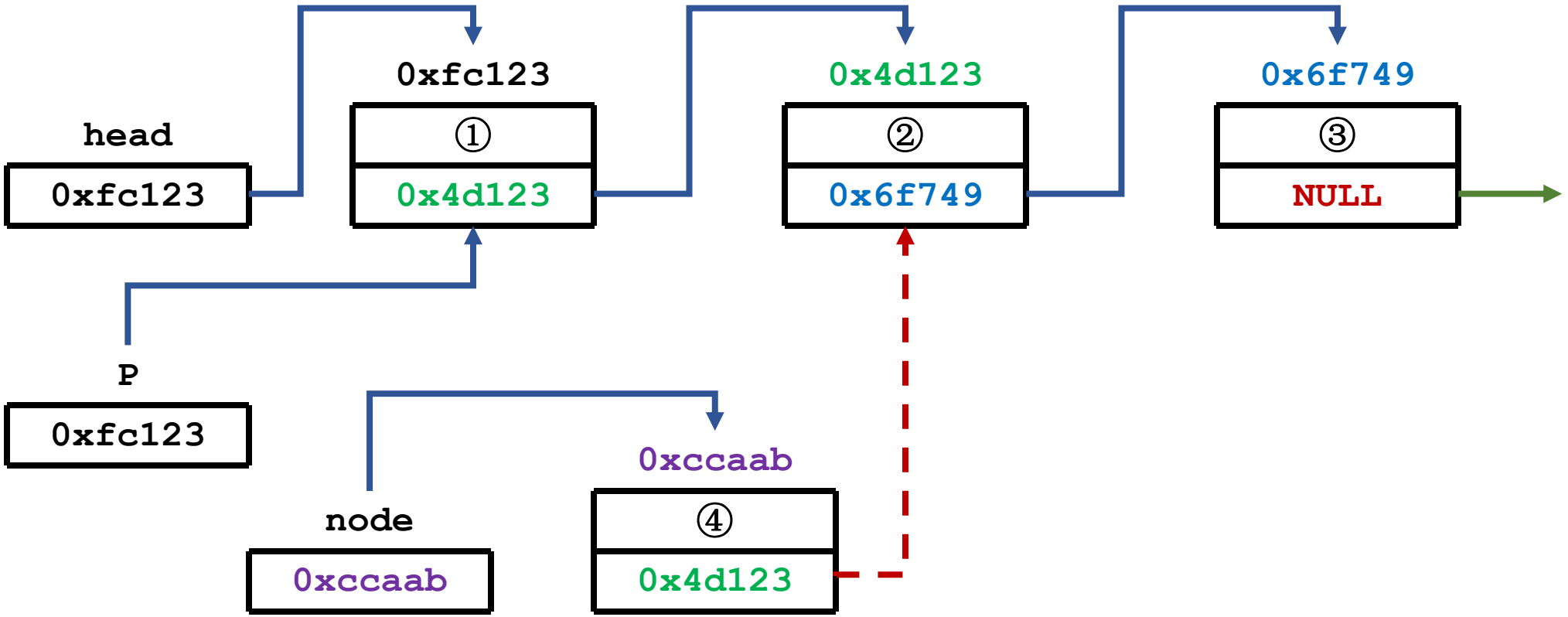
链表：插入元素


head
0xfc123
0xfc123
①
0xccaab
0xccaab
④
0x4d123
0x4d123
②
0x6f749
0x6f749
③
NULL
P
0xfc123
node
0xccaab

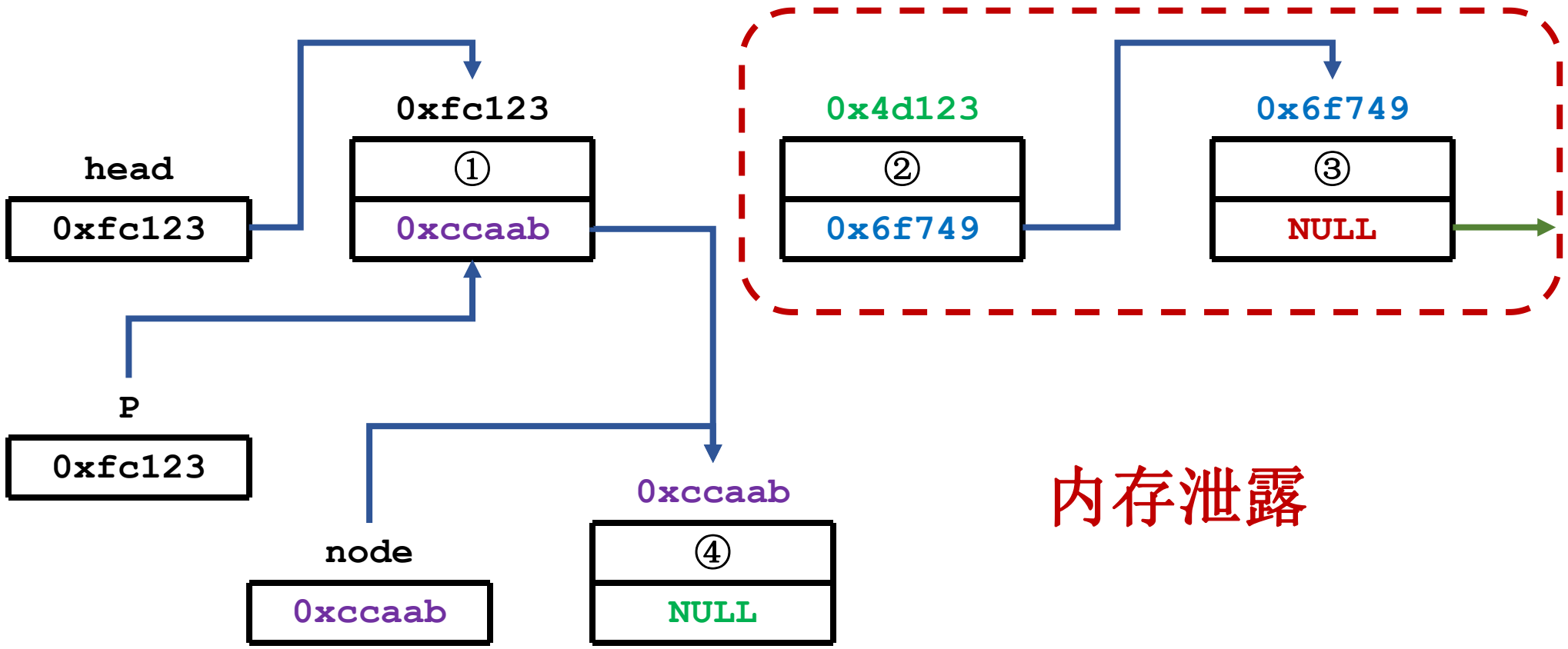
链表：错误插入


0xfc123
0x4d123
0x6f749
head
0xfc123
①
0x4d123
②
0x6f749
③
NULL
P
0xfc123
0xccaab
node
0xccaab
④
NULL

链表：错误插入

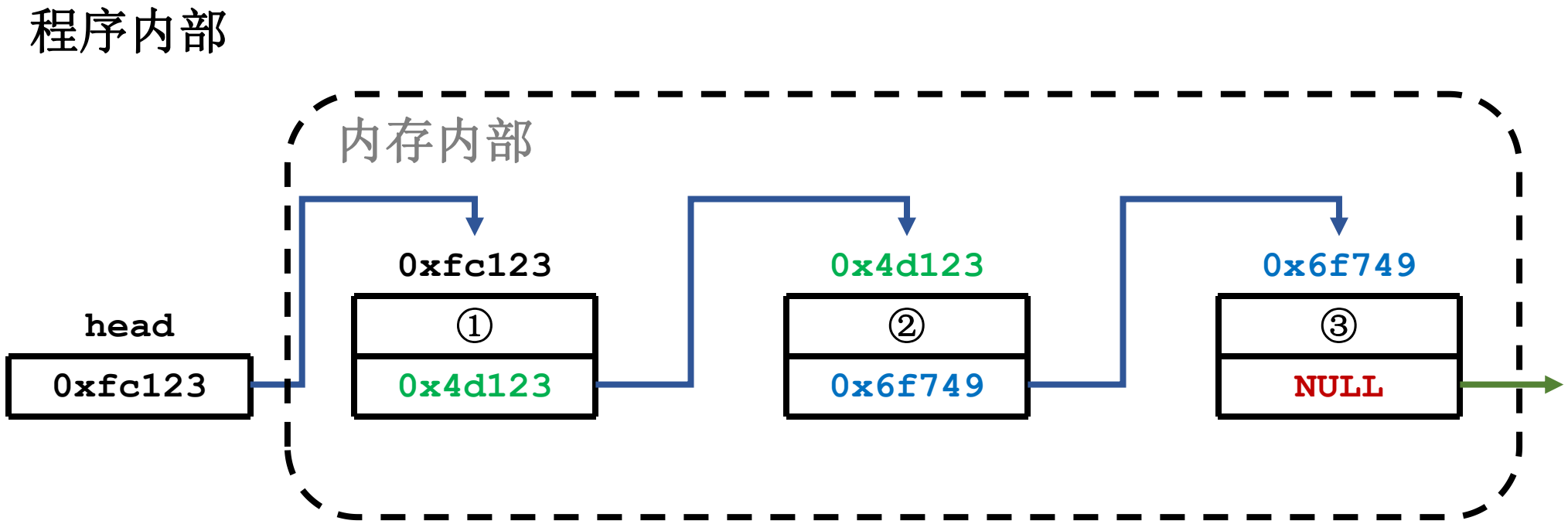

0xfc123
0x4d123
0x6f749
head
0xfc123
①
0x4d123
②
0x6f749
③
NULL
P
0xfc123
node
0xccaab
0xccaab
④
0x4d123

链表：错误插入

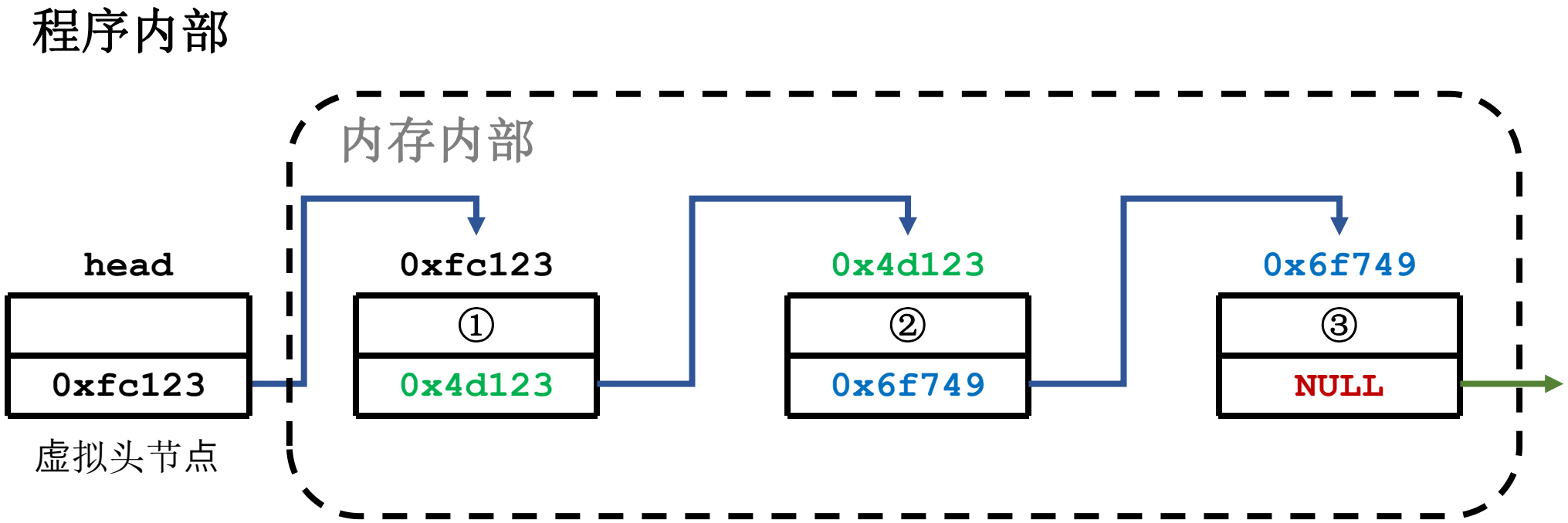

0xfc123
head
0xfc123
①
0xccaab
0x4d123
②
0x6f749
0x6f749
③
NULL
P
0xfc123
node
0xccaab
0xccaab
④
NULL

内存泄露

链表：无头链表


程序内部
内存内部
head
0xfc123
0xfc123
①
0x4d123
0x4d123
②
0x6f749
0x6f749
③
NULL

链表：有头链表

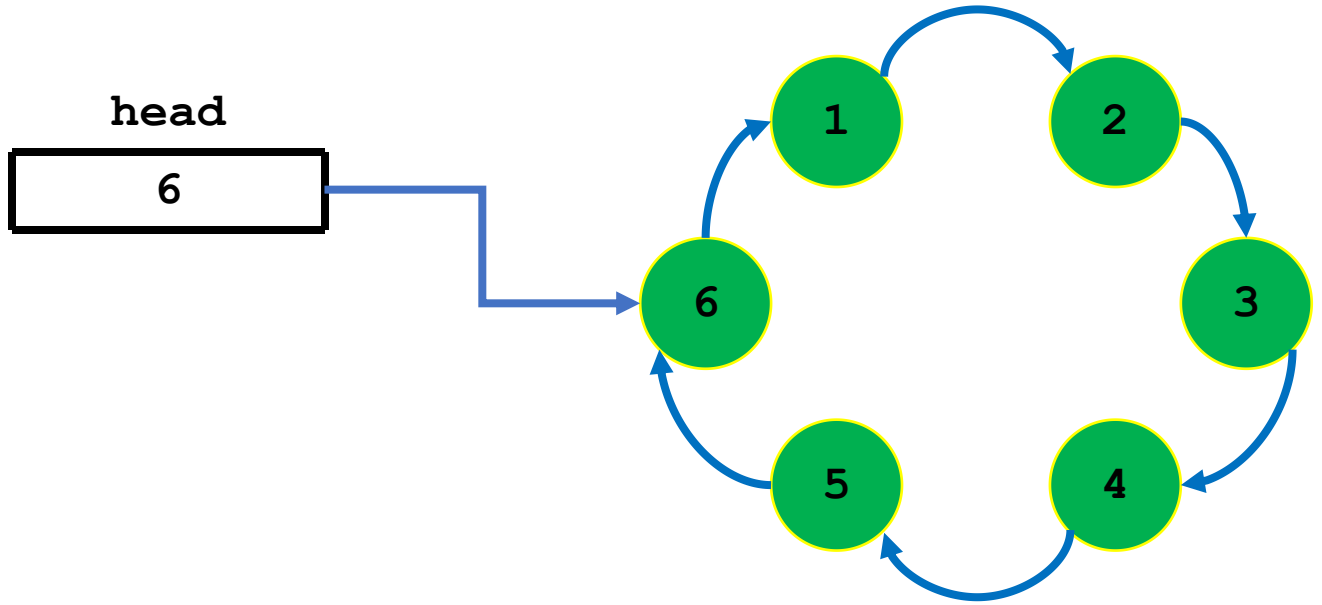

程序内部
内存内部
head
0xfc123
虚拟头节点
0xfc123
①
0x4d123
0x4d123
②
0x6f749
0x6f749
③
NULL


1. vim
vim ⌘1
bash ⌘2
bash ⌘3
39 }
40
41 Node *insert_maintain(Node *root) {
42 if (!hasRedChild(root)) return root;
43 if (root->lchild->color == RED && root->rchild->color == RED) {
44 if (!hasRedChild(root->lchild) && !hasRedChild(root->rchild)) return root;
45 root->color = RED;
46 root->lchild->color = root->rchild->color = BLACK;
47 return root;
48 }
49 if (root->lchild->color == RED) {
50 if (!hasRedChild(root->lchild)) return root;
51
52
53 } else {
54 if (!hasRedChild(root->rchild)) return root;
55
56 }
57
58 }
59
60
61 Node *__insert(Node *root, int key) {
62 if (root == NIL) return getNewNode(key);
<-6班 资 料 /X.现 场 撸 代 码 /15.RBT.cpp [FORMAT=unix] [TYPE=CPP] [POS=54,30][62%] 21/09/19 - 20:21

链表：代码演示

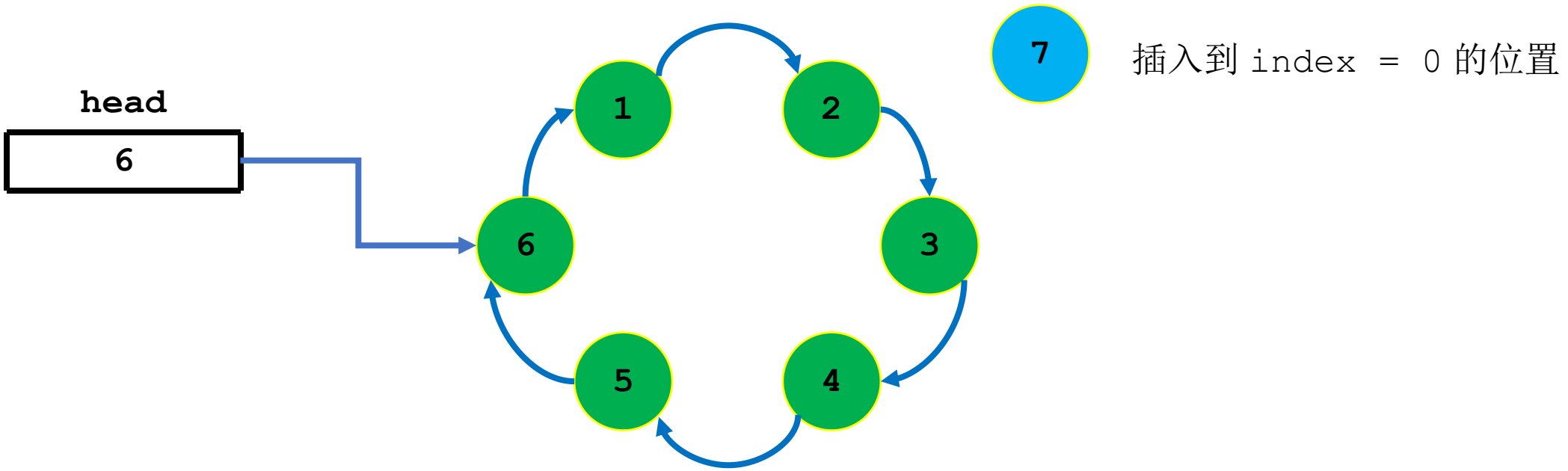
三. 循环链表 和 双向链表

单向循环链表



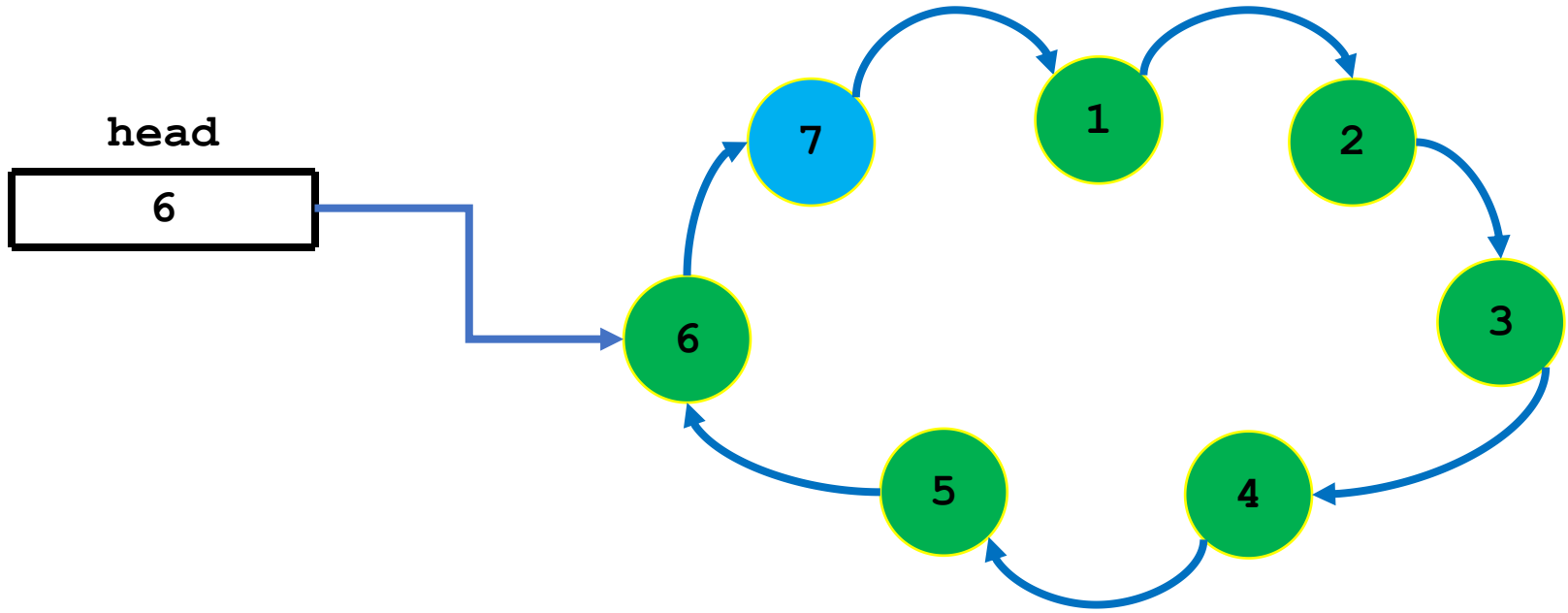
把 head 看做整个单向循环链表的尾节点

单向循环链表-插入



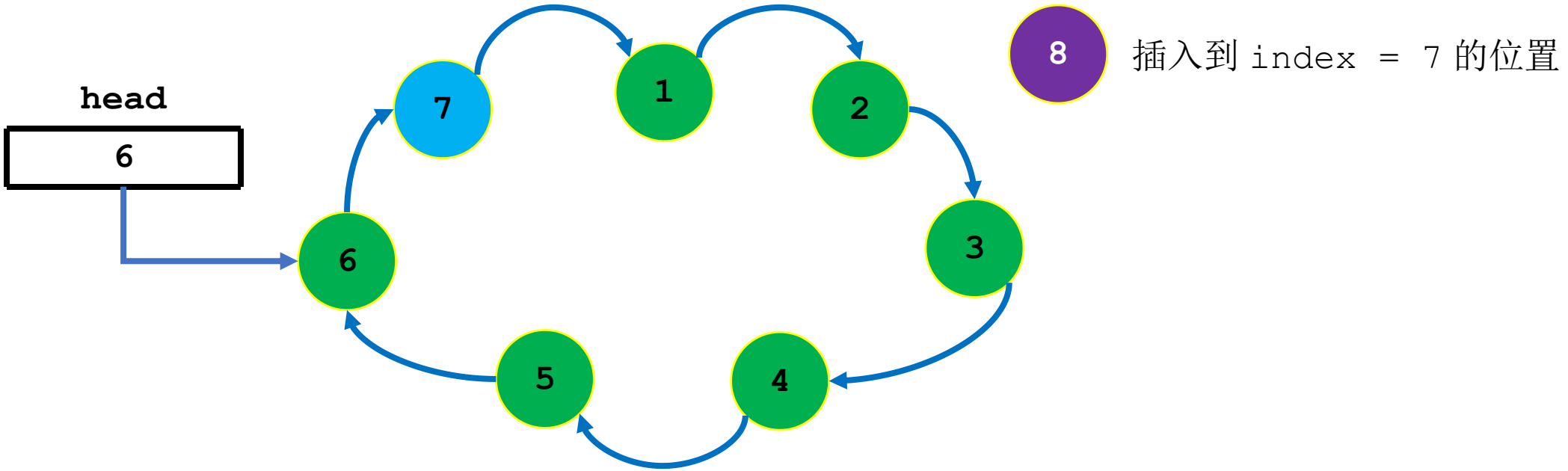
把 head 看做整个单向循环链表的尾节点

单向循环链表-插入



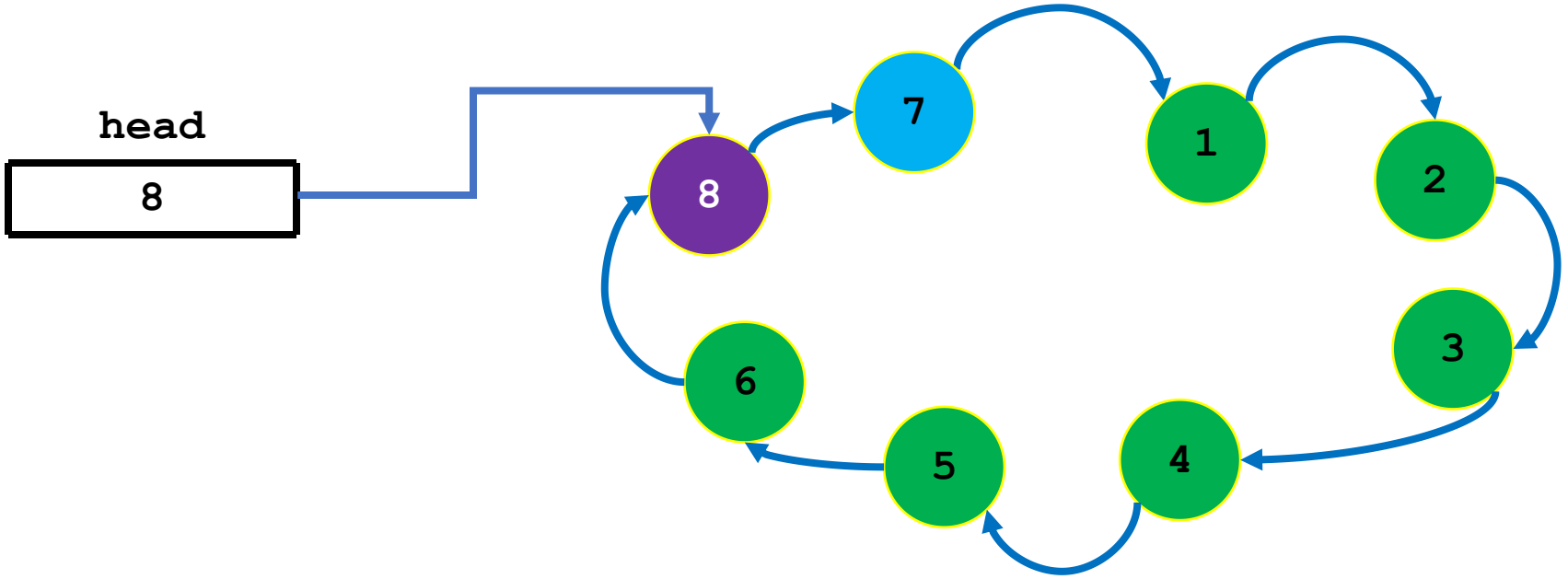
把 head 看做整个单向循环链表的尾节点

单向循环链表-插入



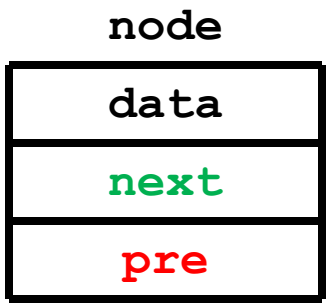
把 head 看做整个单向循环链表的尾节点

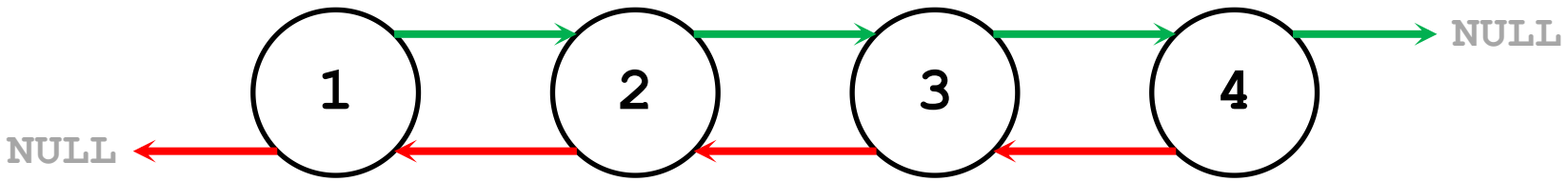
单向循环链表-插入

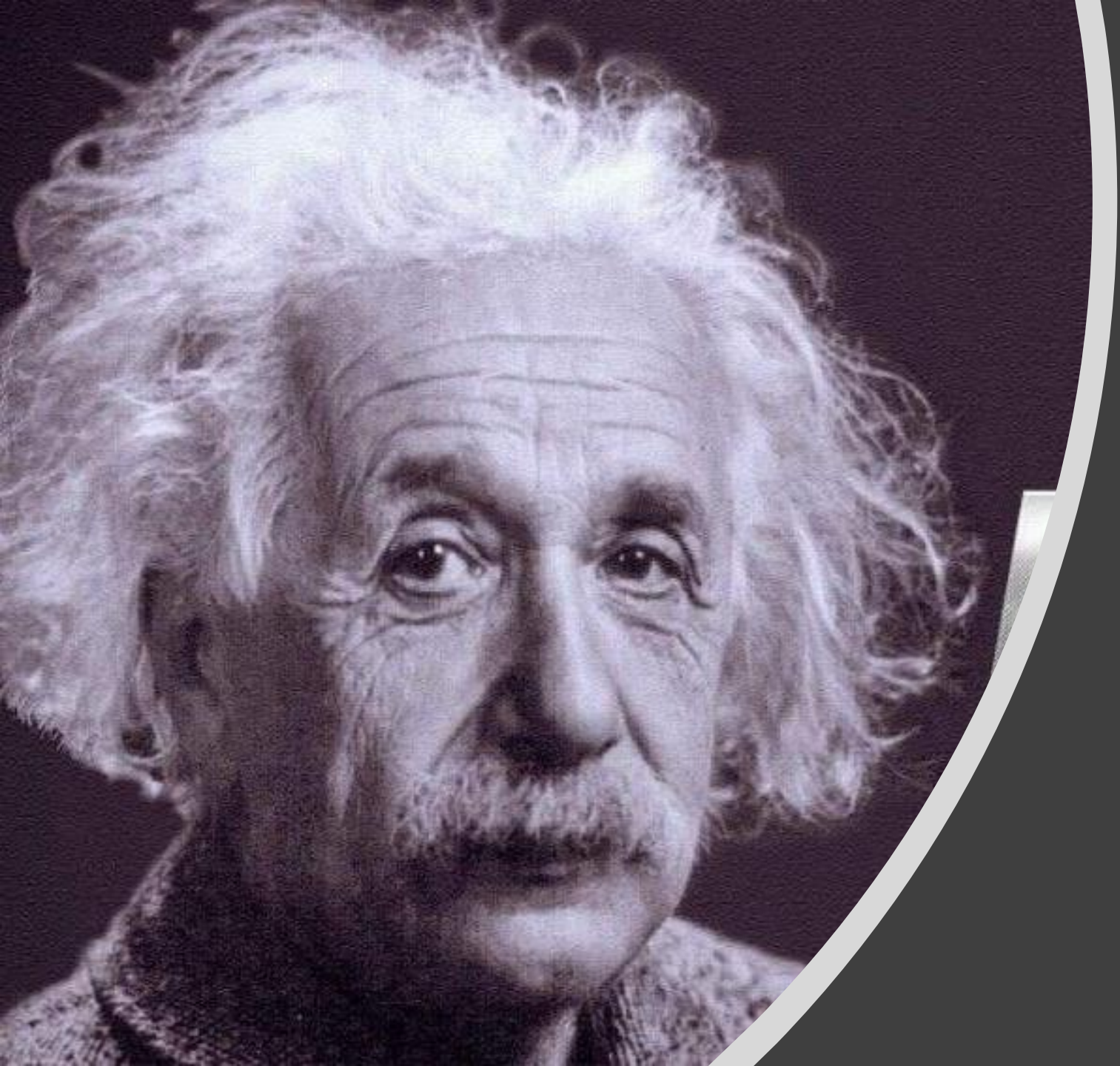


把 head 看做整个单向循环链表的尾节点

三. 双向链表







为什么
会出一样的题目？