栈和队列

胡船长

初航我带你，远航靠自己

本章习题

1-应试. HZOJ-595：程序调用关系

2-应试. HZOJ-838：2020年数据结构41题

3-应试. Leetcode-844：比较退格的字符串

4-校招. HZOJ-263：火车进站

5-校招. Leetcode-946：验证栈序列

6-校招. HZOJ-265：括号画家

7-校招. Leetcode-622：设计循环队列

8-竞赛. HZOJ-266：表达式计算

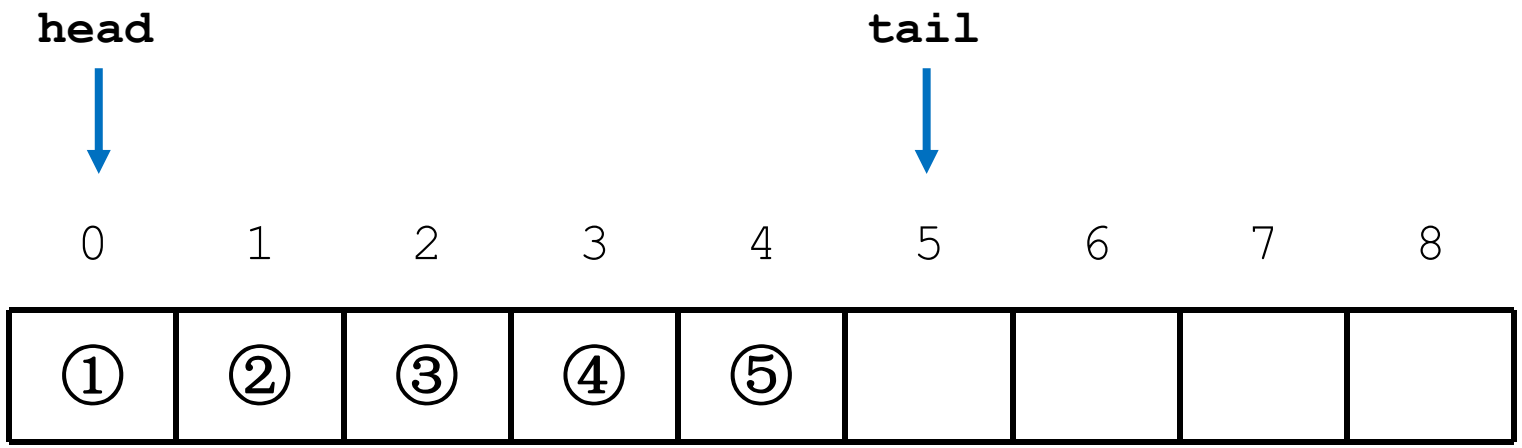
本期内容

一. 队列：结构讲解 & 代码演示

二. 栈：结构讲解 & 代码演示

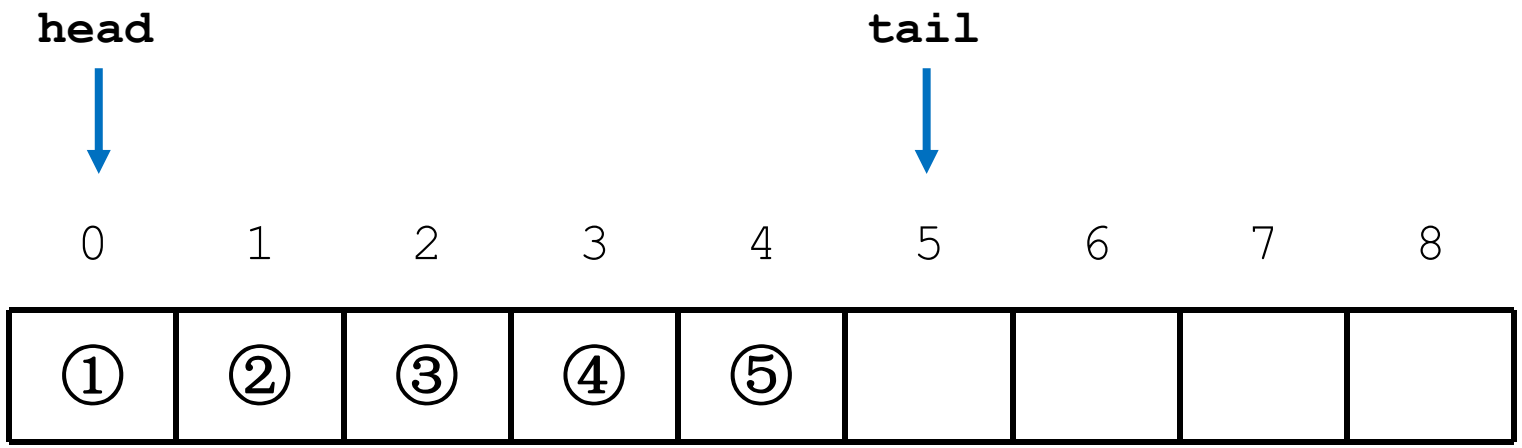
一. 队列：结构讲解 & 代码演示

队列：结构定义



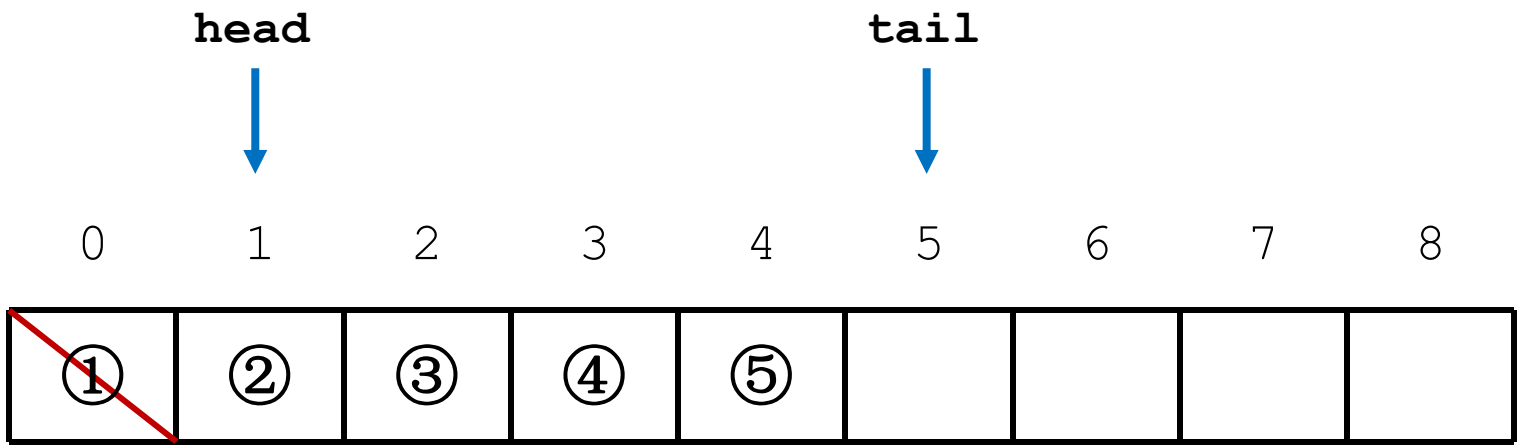
1、size = 9
2、head = 0
3、tail = 5

队列：出队

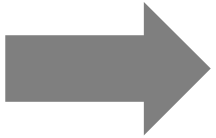


1、size = 9
2、head = 0
3、tail = 5

队列：出队



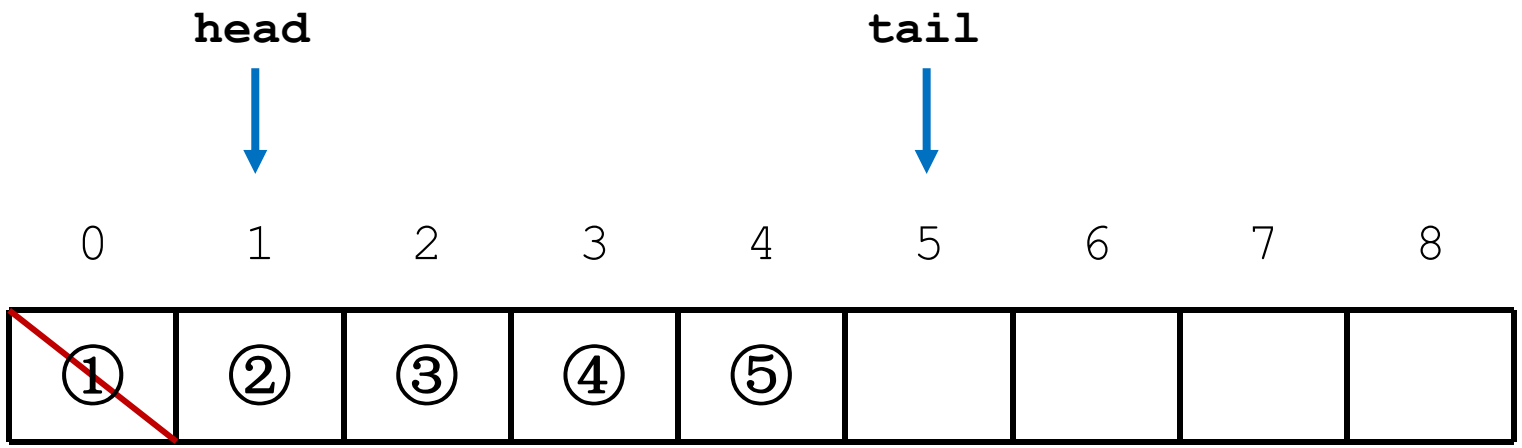
1、size = 9
2、head = 0
3、tail = 5



1、size = 9
2、head = 1
3、tail = 5

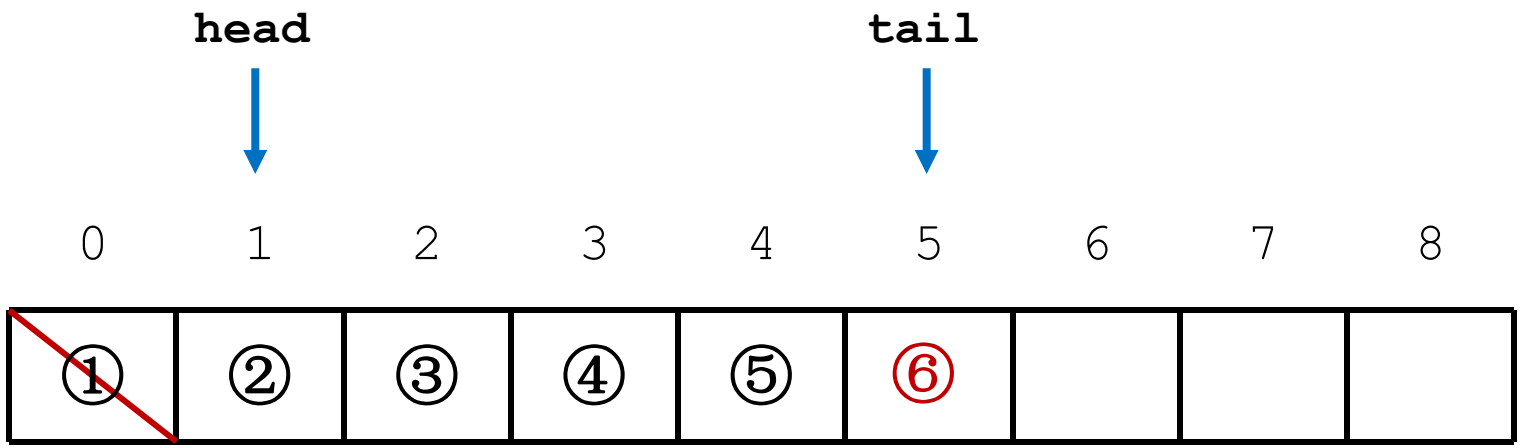
队列：入队

⑥



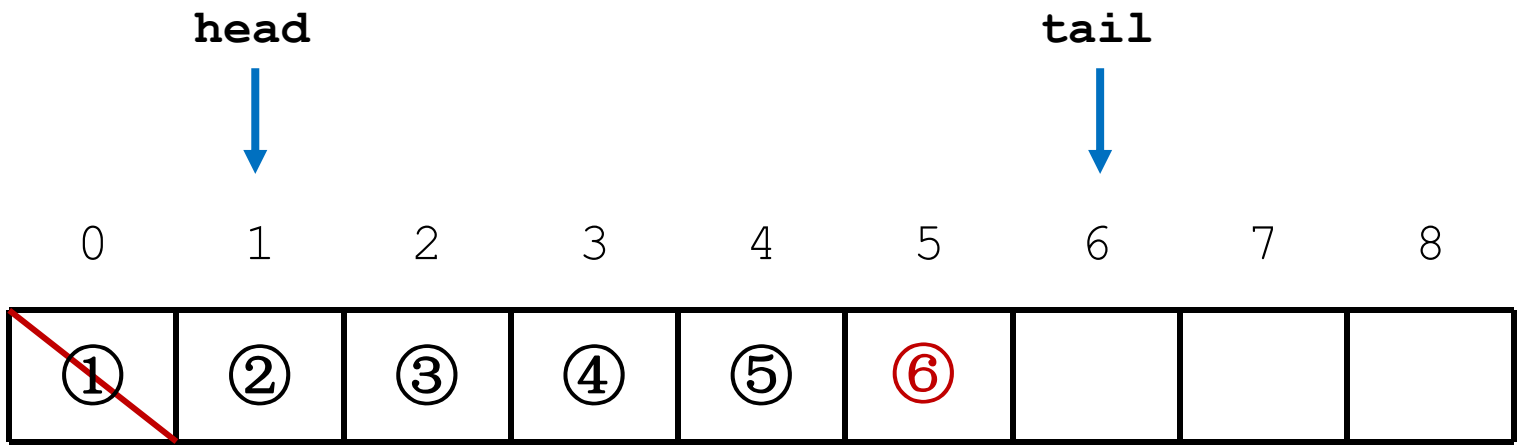
1、size = 9
2、head = 1
3、tail = 5

队列：入队

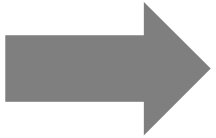


1、size = 9
2、head = 1
3、tail = 5

队列：入队

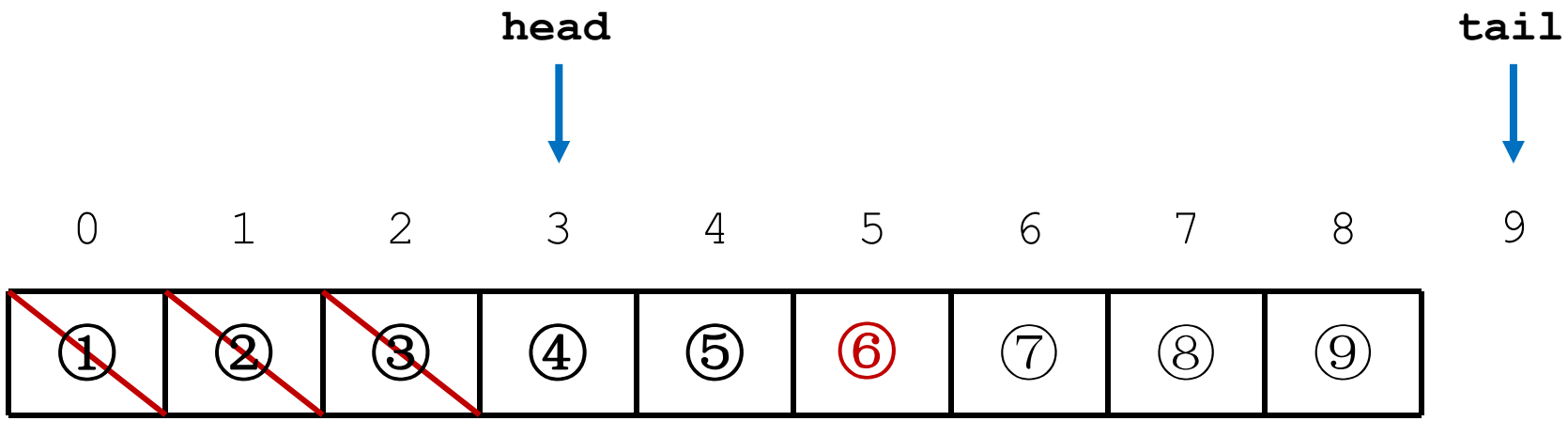


1、size = 9
2、head = 1
3、tail = 5



1、size = 9
2、head = 1
3、tail = 6

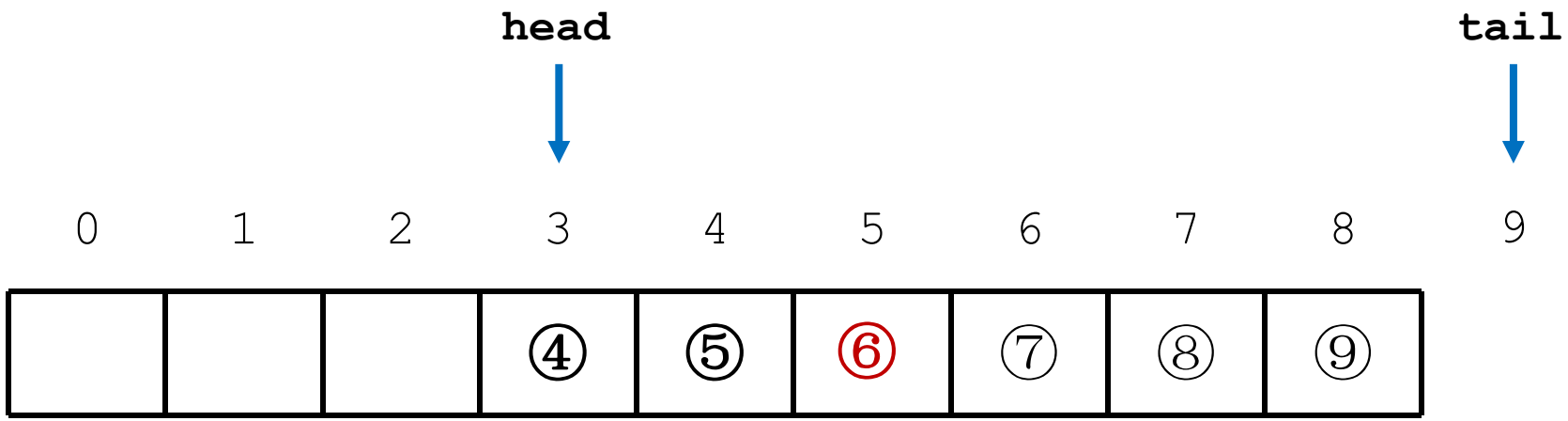
队列：假溢出



插入 ⑩ ？

1、size = 9
2、head = 3
3、tail = 9

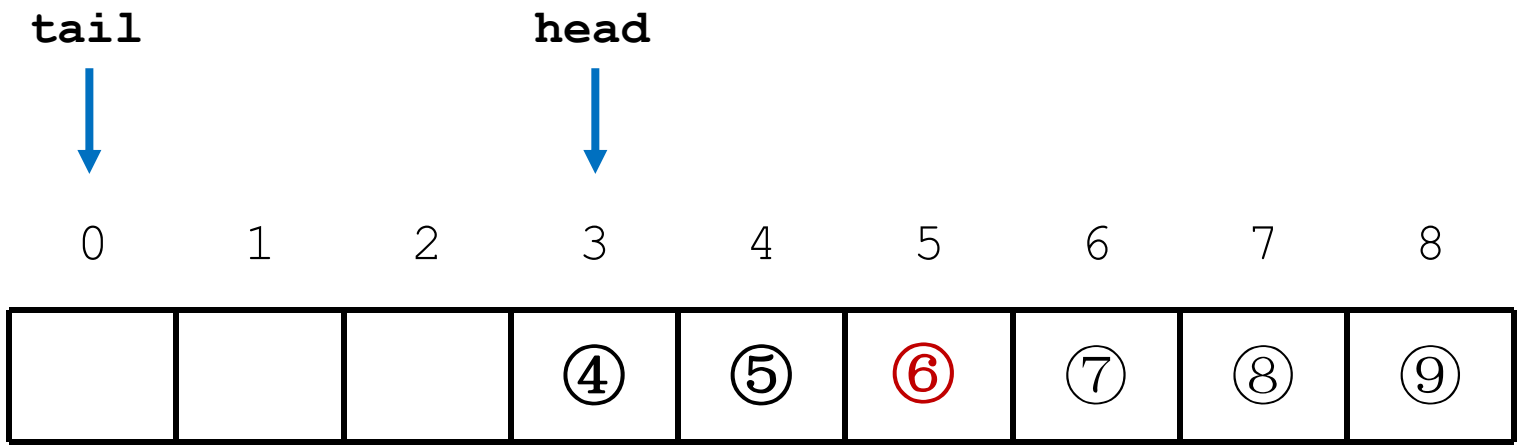
队列：假溢出



插入 ⑩ ？

1、size = 9
2、head = 3
3、tail = 9

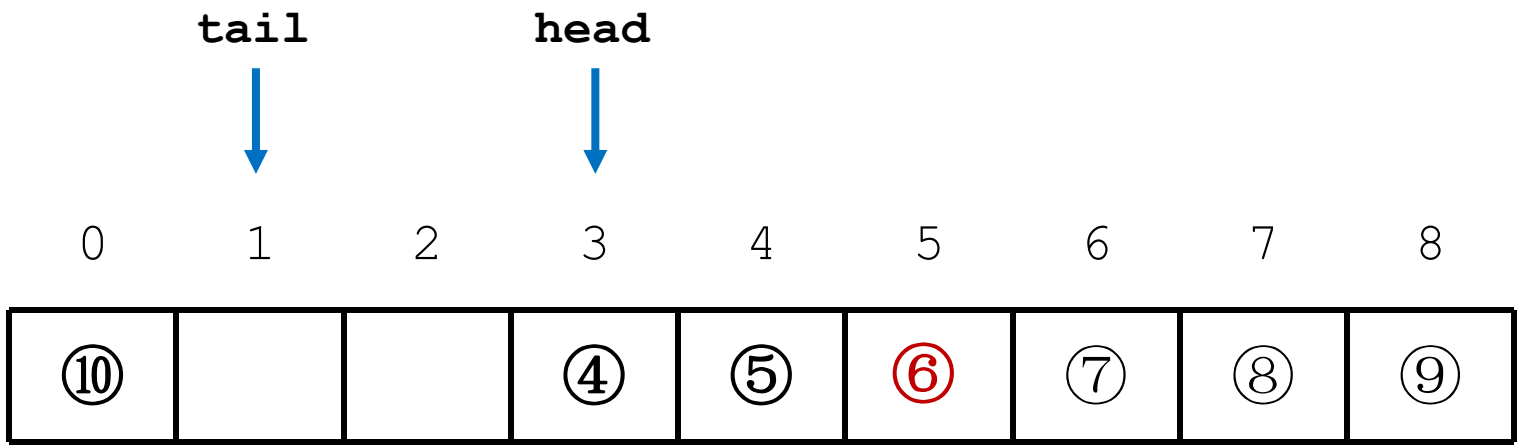
队列：循环队列



插入 ⑩ ？

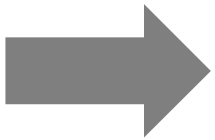
1、size = 9
2、head = 3
3、tail = 0
4、count = 6

队列：循环队列



插入 ⑩ ？

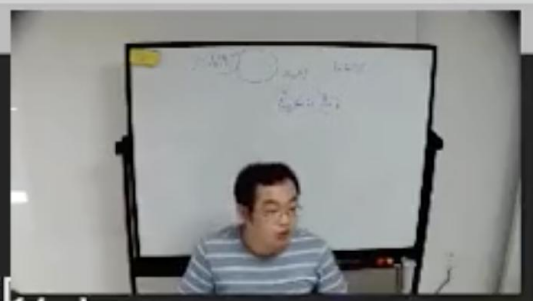
1、size = 9
2、head = 3
3、tail = 0
4、count = 6



1、size = 9
2、head = 3
3、tail = 1
4、count = 7

1. vim

vim ⌘1 bash ⌘2 bash ⌘3



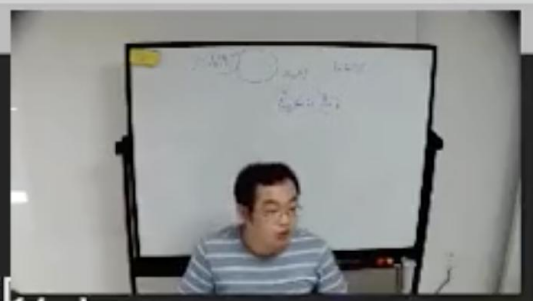
```
39 }
40
41 Node *insert_maintain(Node *root) {
42     if (!hasRedChild(root)) return root;
43     if (root->lchild->color == RED && root->rchild->color == RED) {
44         if (!hasRedChild(root->lchild) && !hasRedChild(root->rchild)) return root;
45         root->color = RED;
46         root->lchild->color = 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);
```

队列：代码演示-顺序表实现

<-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

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);
```

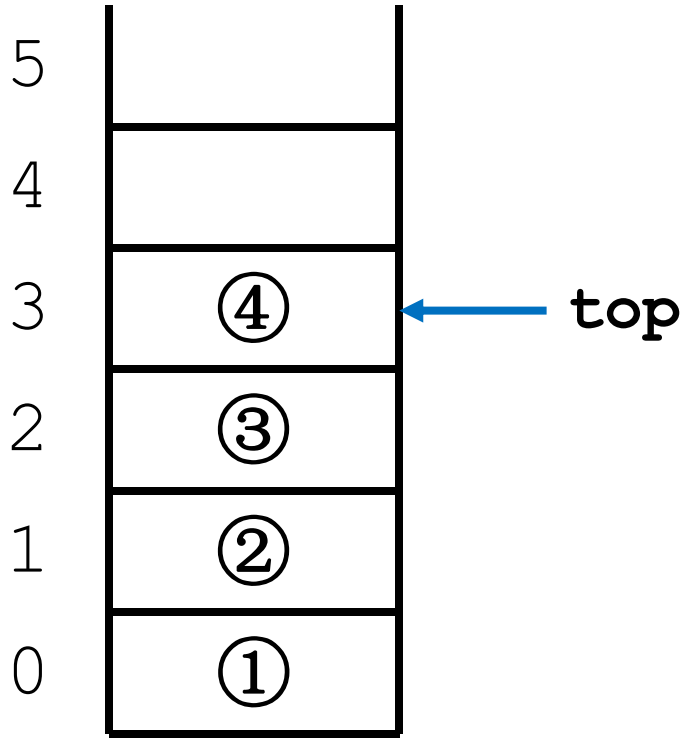
队列：代码演示-链表实现

<-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

二. 栈：结构讲解 & 代码演示

栈：结构定义

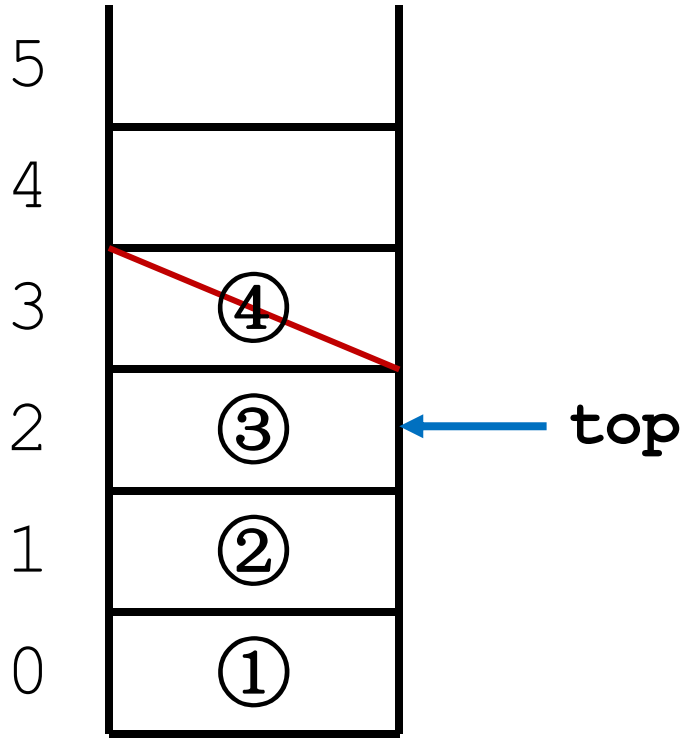
1、size = 5
2、top = 3



栈：出栈

1、size = 5
2、top = 3

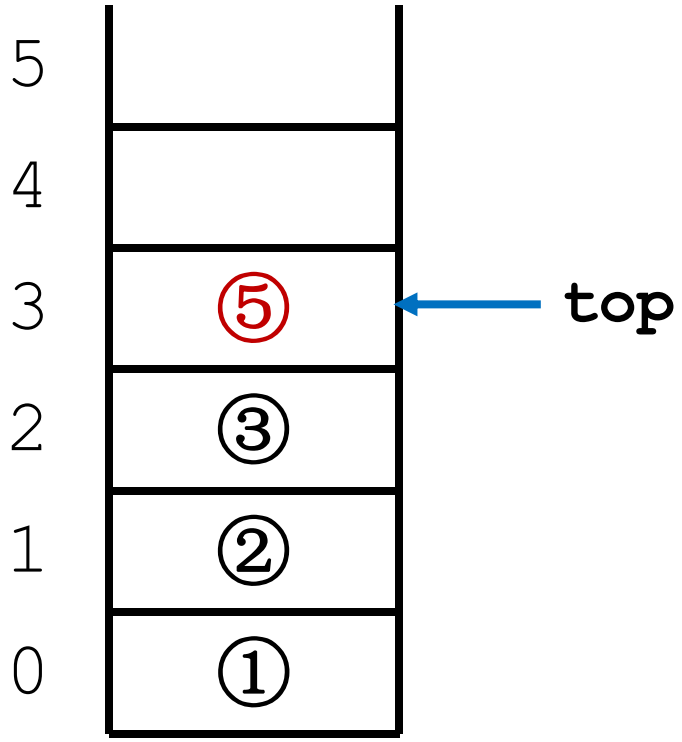
1、size = 5
2、top = 2



栈：入栈

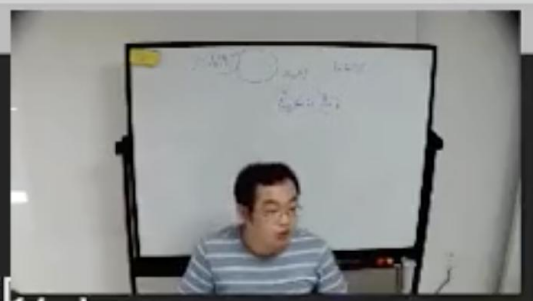
1、size = 5
2、top = 2

1、size = 5
2、top = 3



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);
```

栈：代码演示

<-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

Leetcode-20： 括号匹配

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

The brackets must close in the correct order, "()" and "()[]{}" are all valid but "(]" and "([)]" are not.

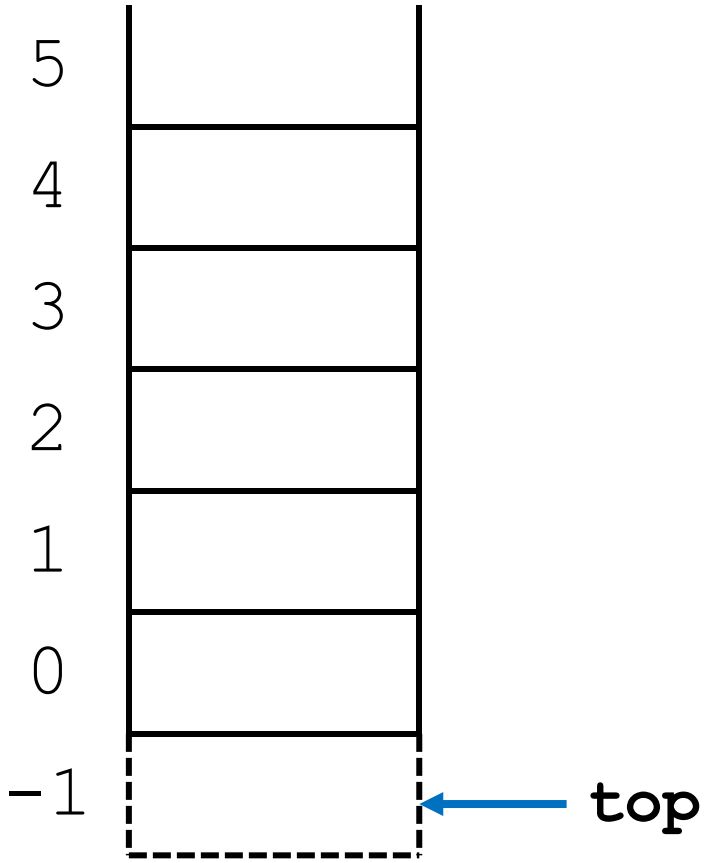
Leetcode-20： 括号匹配

用栈模拟括号匹配的过程

1. 遇到『左括号』就进栈
2. 遇到『右括号』就和栈顶的『左括号』做匹配，如果**成功**，栈顶元素出栈，如果**失败**，说明括号序列不合法

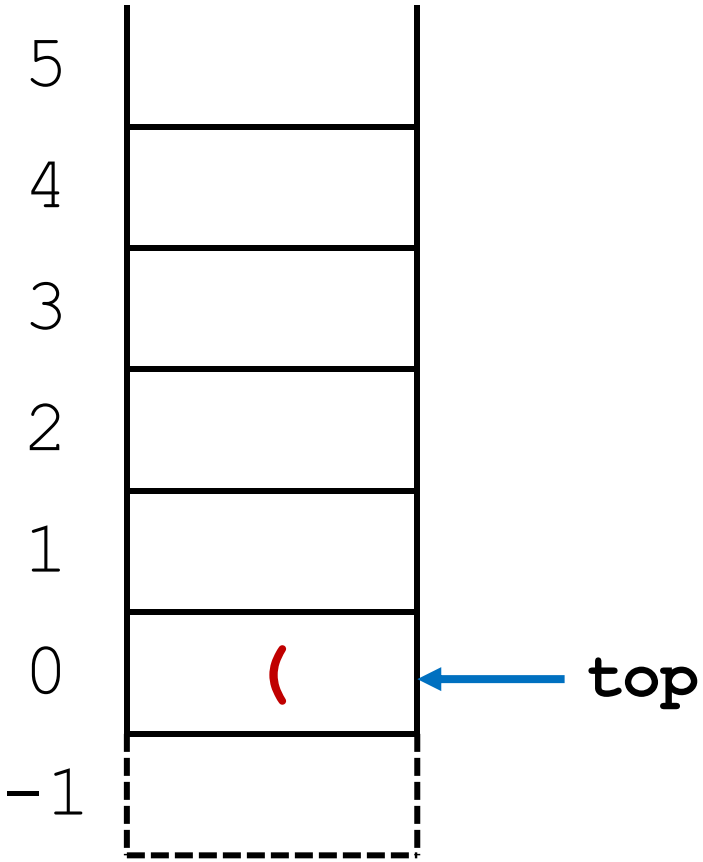
Leetcode-20： 括号匹配

({ }) [( ) { }]



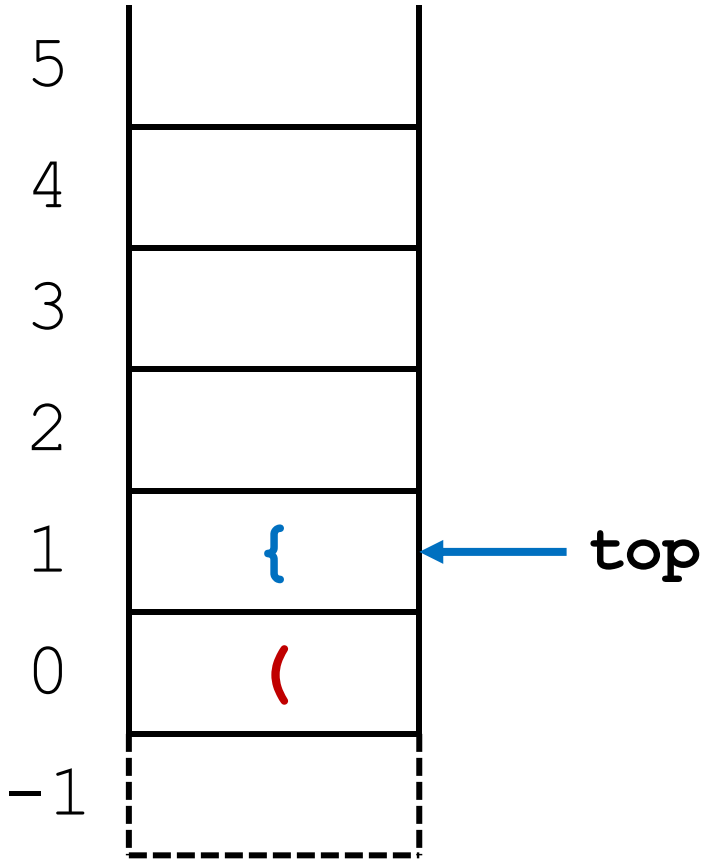
Leetcode-20： 括号匹配

({ }) [( ) { }]



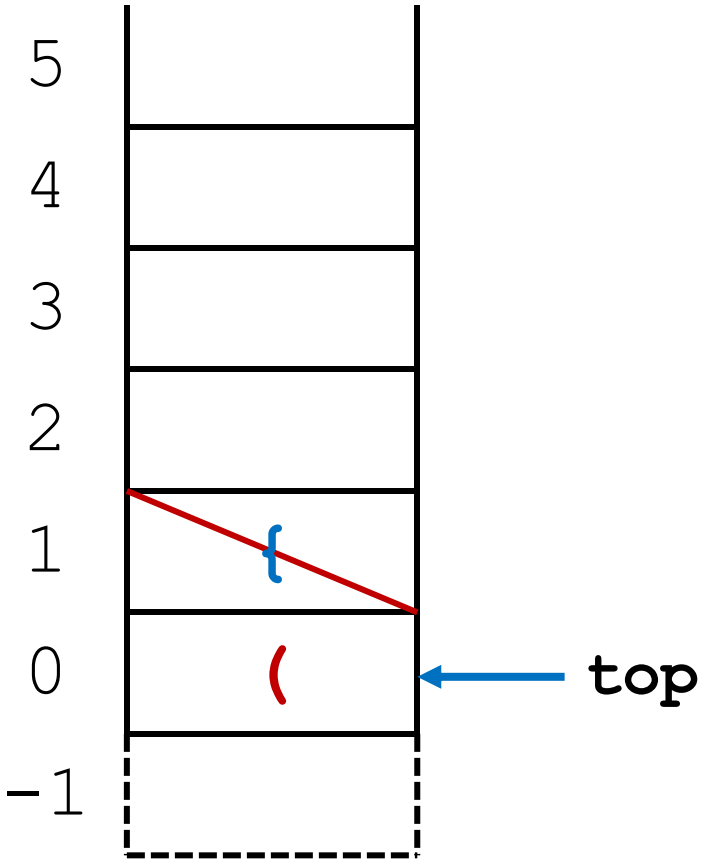
Leetcode-20： 括号匹配

({ }) [( ) { }]

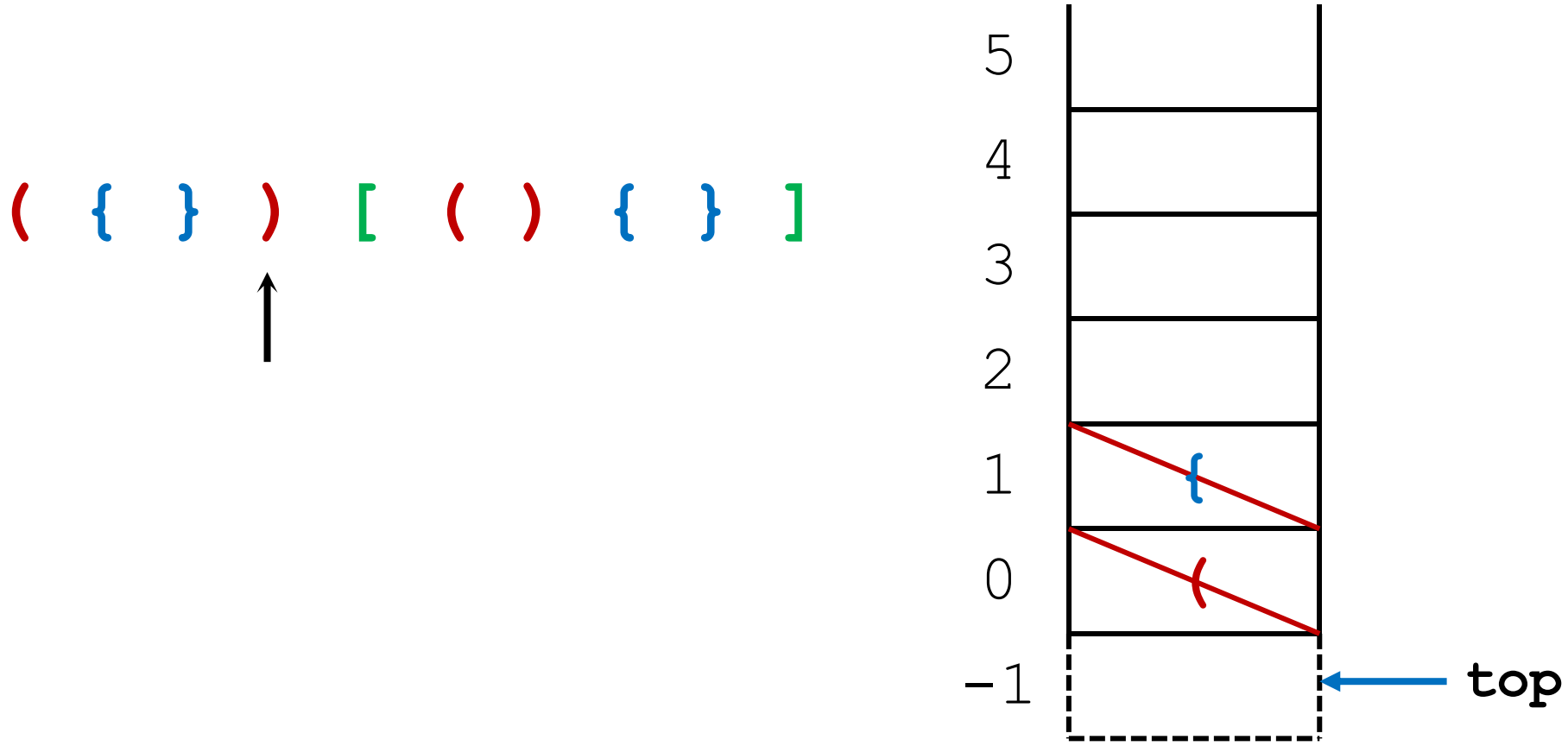


Leetcode-20： 括号匹配

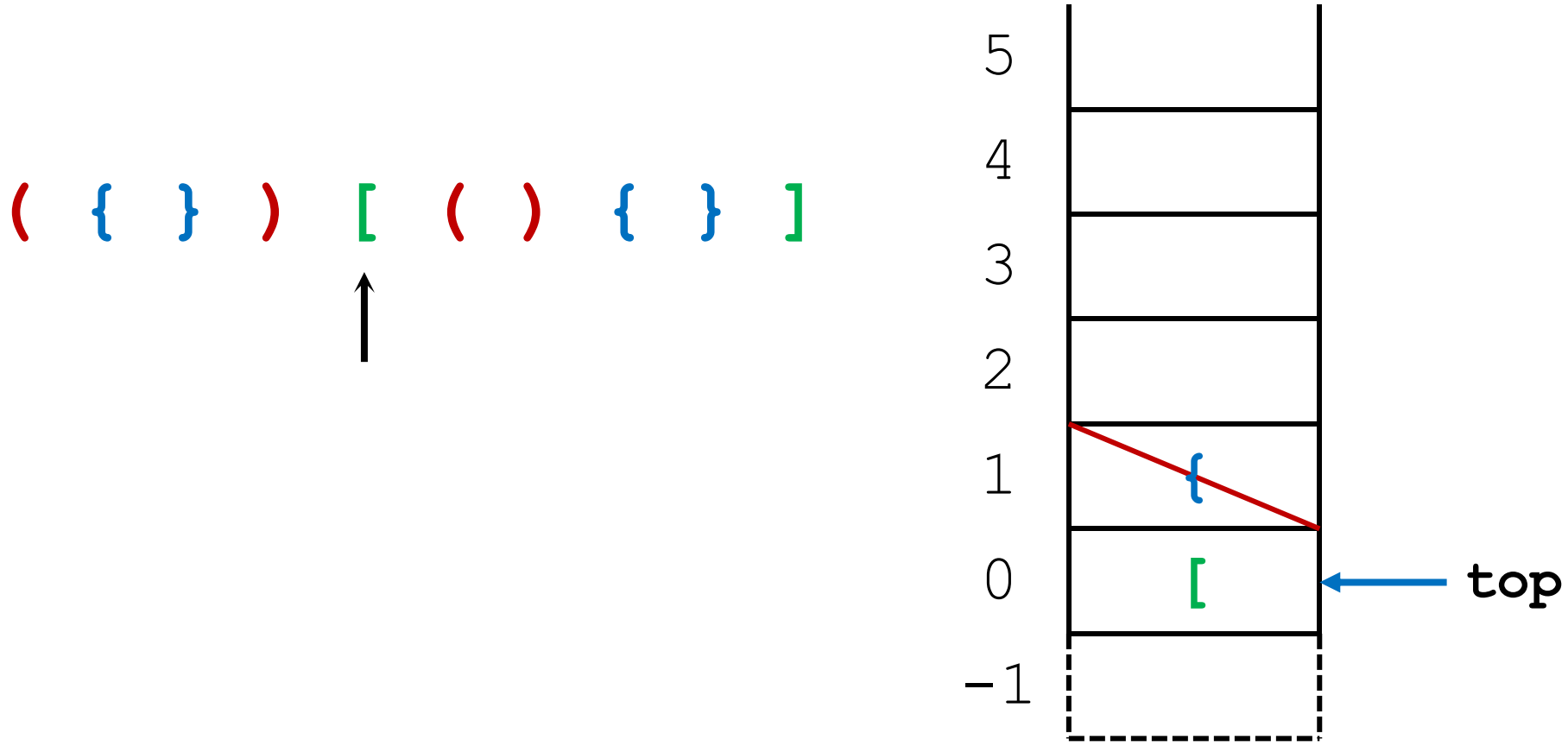
({ }) [( ) { }]



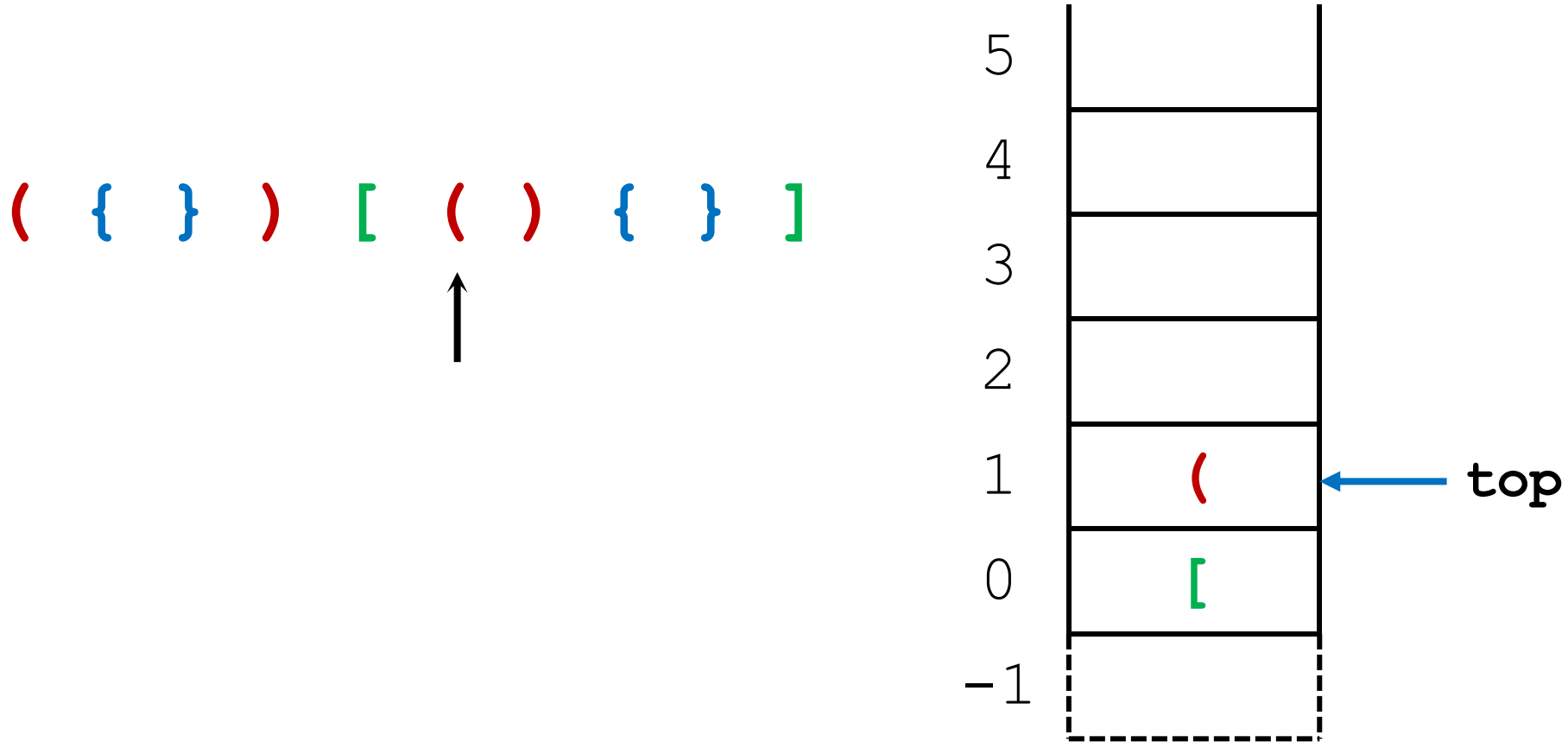
Leetcode-20： 括号匹配



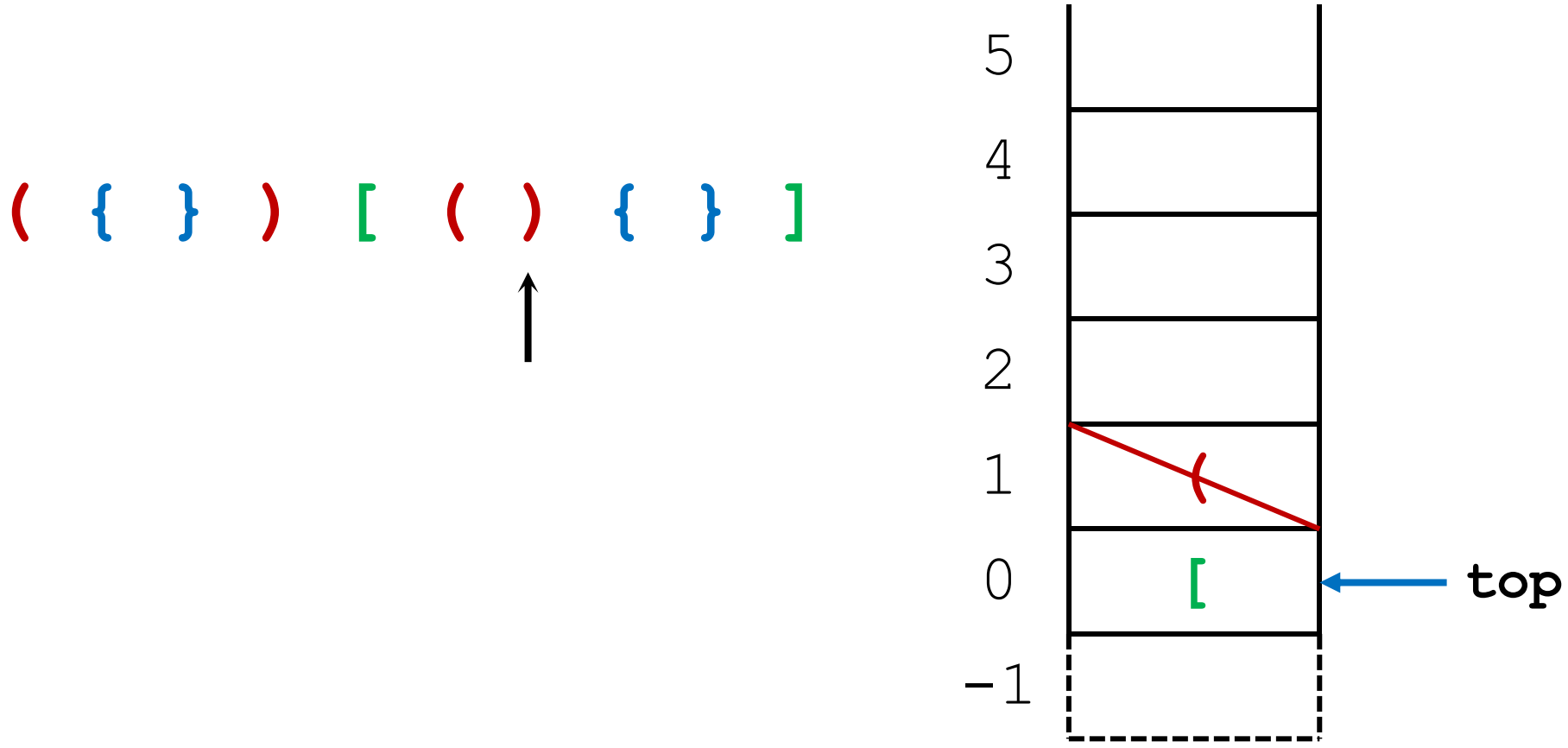
Leetcode-20： 括号匹配



Leetcode-20： 括号匹配

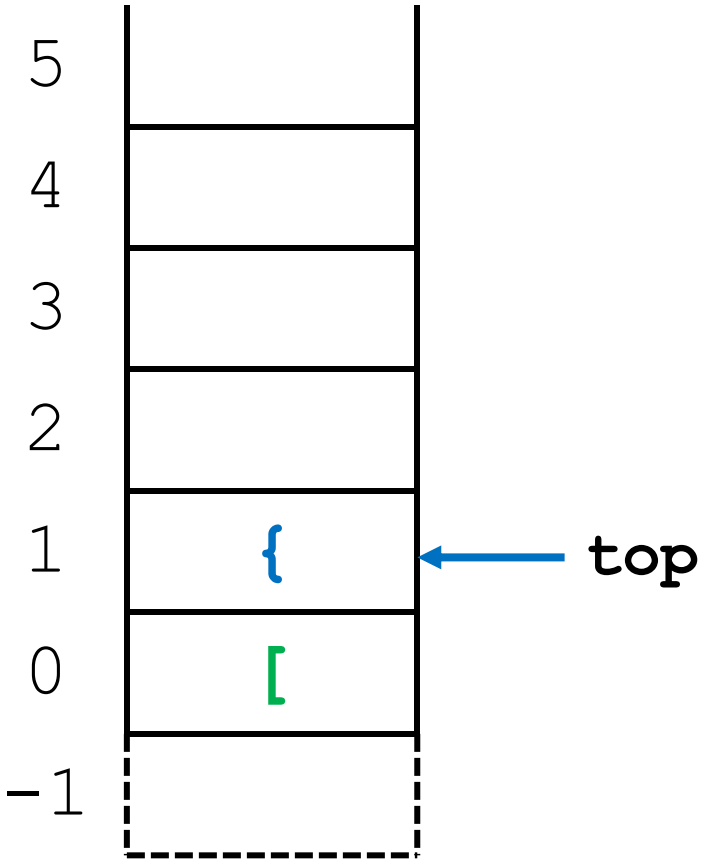


Leetcode-20： 括号匹配



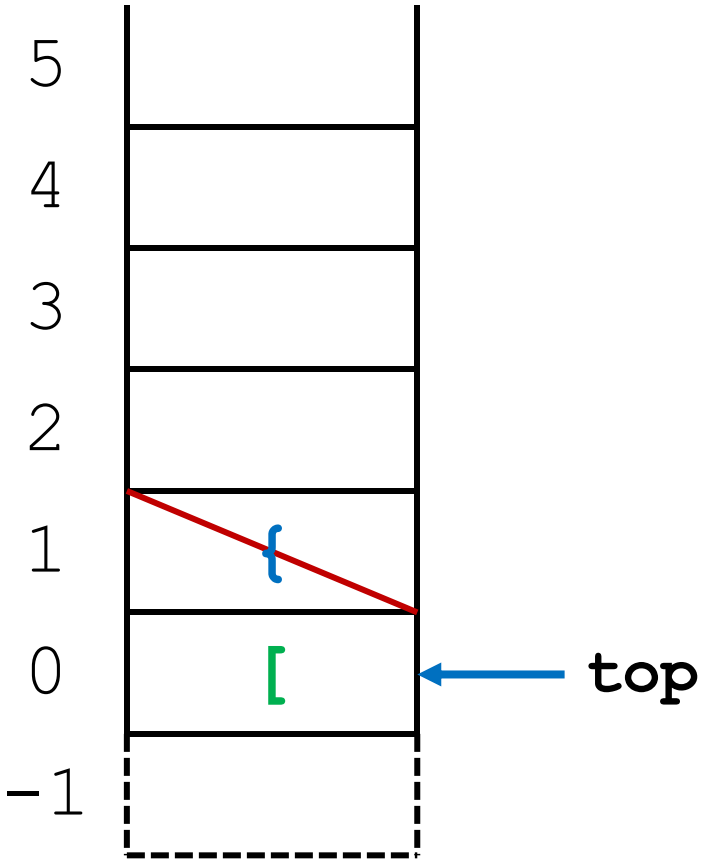
Leetcode-20： 括号匹配

({ }) [( ) { }]

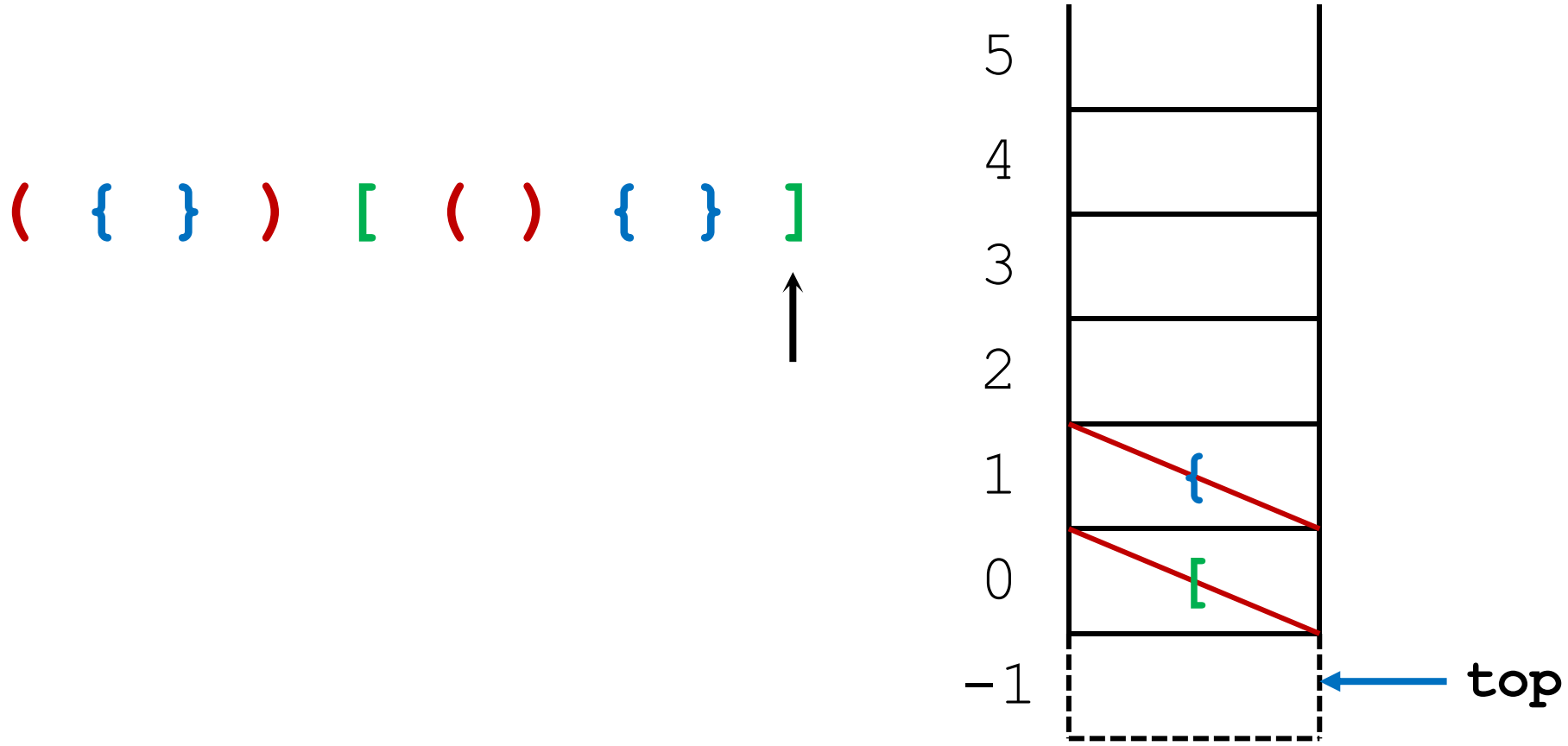


Leetcode-20： 括号匹配

({ }) [( ) { }]

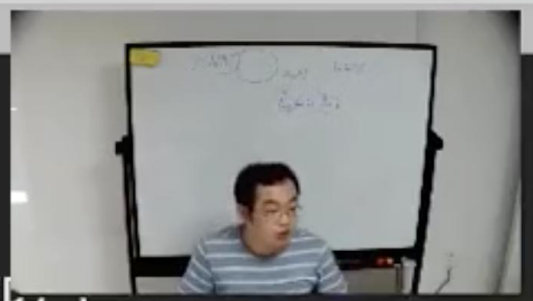


Leetcode-20： 括号匹配



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);
```

Leetcode-20 括号匹配：代码演示

<-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

栈：深入理解

思考：

问题简化成只有一种括号，怎么做?

想到用栈的同学，在简化问题中，能不能不用栈?

栈：深入理解

1、(((()())())())

2、((()))()()

3、(()()()

栈：深入理解

1、(((()))())　　**TRUE**

2、(()))()　　**FALSE**

3、(()()　　**FALSE**

栈：深入理解

结论：

1、在任意一个位置上，左括号数量 >= 右括号数量

2、在最后一个位置上，左括号数量 == 右括号数量

3、程序中只需要记录左括号数量和右括号数量即可

栈：深入理解

```
1  bool isValid(char *s) {
2      int32_t lnum = 0, rnum = 0;
3      int32_t len = strlen(s);
4      for (int32_t i = 0; i < len; i++) {
5          switch (s[i]) {
6              case '(' : ++lnum; break;
7              case ')' : ++rnum; break;
8              default : return false;
9          }
10         if (lnum >= rnum) continue;
11         return false;
12     }
13     return lnum == rnum;
14 }
```

存在问题：

程序能不能更优化？

思考：

rnum 变量一定是需要的么？

栈：深入理解

```
1  bool isValid(char *s) {
2      int32_t lnum = 0;
3      int32_t len = strlen(s);
4      for (int32_t i = 0; i < len; i++) {
5          switch (s[i]) {
6              case '(' : ++lnum; break;
7              case ')' : --lnum; break;
8              default : return false;
9          }
10         if (lnum >= 0) continue;
11         return false;
12     }
13     return lnum == 0;
14 }
```

栈：深入理解

思考：

1、我们获得了怎样新的思维方式?
2、+1 可以等价为『进』，-1可以等价为『出』
3、一对 ( ) 可以等价为一个完整的事件
4、(( )) 可以看做事件与事件之间的完全包含关系
5、由括号的等价变换，得到了一个新的数据结构

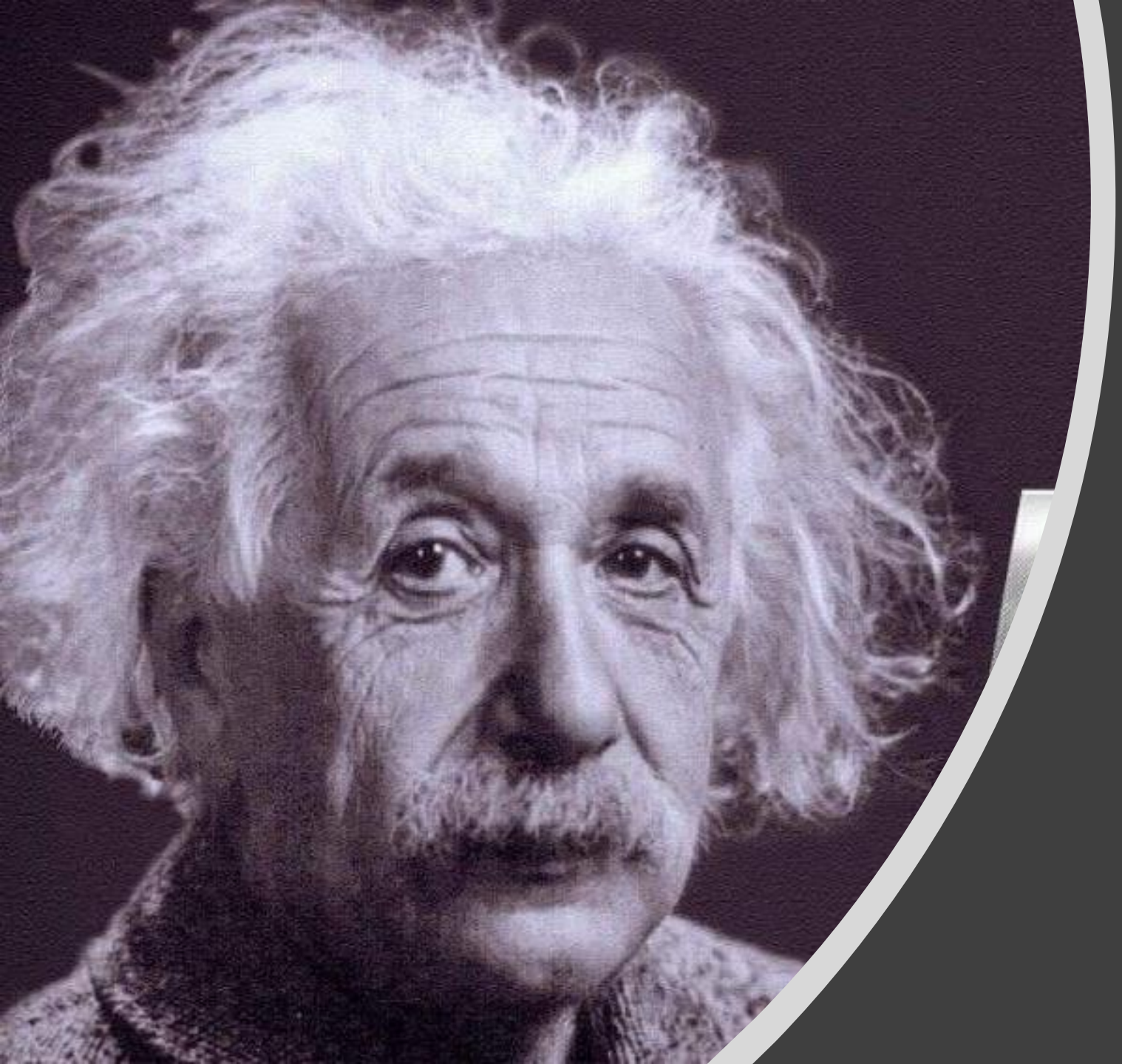
栈：深入理解

栈

可以处理具有完全包含关系的问题

栈和队列的应用

栈	树的深度遍历、深度优先搜索(图算法基础)
队列(循环)	树的层序遍历，广度优先搜索(图算法基础)
单调栈	临近最大(小)值
单调队列	区间最大(小)值



为什么

会出一样的题目？