**难度: 简单**

三合一。描述如何只用一个数组来实现三个栈。

你应该实现push(stackNum, value)、pop(stackNum)、isEmpty(stackNum)、peek(stackNum)方法。stackNum表示栈下标，value表示压入的值。

构造函数会传入一个stackSize参数，代表每个栈的大小。

示例1:

输入：

["TripleInOne", "push", "push", "pop", "pop", "pop", "isEmpty"]

[[1], [0, 1], [0, 2], [0], [0], [0], [0]]

输出：

[null, null, null, 1, -1, -1, true]

说明：当栈为空时`pop, peek`返回-1，当栈满时`push`不压入元素。

示例2:

输入：

["TripleInOne", "push", "push", "push", "pop", "pop", "pop", "peek"]

[[2], [0, 1], [0, 2], [0, 3], [0], [0], [0], [0]]

输出：

[null, null, null, null, 2, 1, -1, -1]

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/three-in-one-lcci

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路:**

**题目理解: 首先这个题目太容易让别人懵逼了**

**首先开始创建类时构造函数里面创入的大小是栈的大小 ( stackSize )**

**其次不管是push还是pop还是peek或者isEmpty 的参数 (stackNum)**

**stackNum输入的值,代表的是哪个栈!!!!**

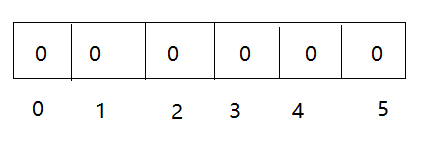
**总共有三个栈 所以 如果 stackNum 为 0 时就代表第一个栈**

**如果 stackNum 为 1 时就代表第二个栈**

1). 由于是一个数组相当于3个栈,所以创建的时候就 乘 3

例如: 创建栈的大小为2 于是 相当于 6

|  |
| --- |
| stack = new int [stackSize \* 3]; |



2). 这个数组相当于3个栈

0 , 3 相当于栈 0

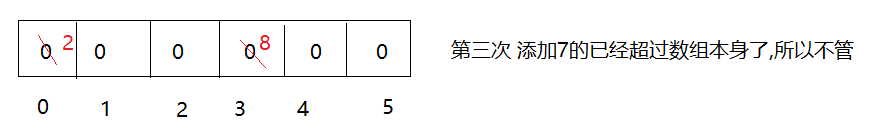
1 ,4 相当于栈 1

2 , 5 相当于栈 2

3). 每次push的时候 需要先判断头指针是否小于数组的长度

如果小于需要添加值,而且需要把top节点加3 ( 根据2的规则 )

例如: 添加栈0 2, 添加栈0 8, 添加栈0 7



4). 每次判断是否为空需要先-3判断是否小于0

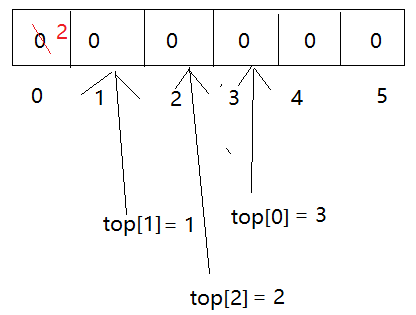
小于 ture 代表为空

不小于 false 代表不为空

5). 每次pop之前需要判断是否为空由于我们是从 0 , 1, 2 分别代表三个栈的

而且我们添加完需要加3, 所以我们每次需要判断是否为空时, 需要 减3 看看是否小于0

例如: 我们先添加栈0 2 ,添加完成后头部的下标就已经在 3了



所以需要pop的时候需要先减3才会到我们添加的节点

当我们pop一次之后, 需要在pop的时候, 会再去判断-3是否小于0

此时top[0] = 0 减3之后就是 -3 所以它是空的返回-1

6). 每次peek的时候, 先判断是否为空 空直接返回-1, 不为空话就把top – 3 的下标给它就可以了,此时不是抛出所以不需要改变原来的top的值

|  |
| --- |
| public class TripleInOne {  //定义一个数组栈  private int[] stack;  //定义一个头指针的数组  private int[] top;  public TripleInOne(int stackSize) {  //定义三栈合一  stack = new int[stackSize \* 3];  // 三个栈的头结点  top = new int[3];  //分别赋值第一次的头结点  // 栈0 top[0] = 0;  // 栈1 top[1] = 1;  // 栈2 top[2] = 2;  for (int i = 0; i < 3; i++) {  top[i] = i;  }  }  public void push(int stackNum, int value) {  //判断是否添加满了  if (top[stackNum] < stack.length) {  //赋值  stack[top[stackNum]] = value;  //头结点+3  top[stackNum] = top[stackNum] + 3;  }  }  public int pop(int stackNum) {  //判断是否为空  if (isEmpty(stackNum)) {  return -1;  }  //原本的top[stackNum]减3  top[stackNum] = top[stackNum] - 3;  //返回  return stack[(top[stackNum])];  }  public int peek(int stackNum) {  //判断是否为空  if (isEmpty(stackNum)) {  return -1;  }  //把下标给它但是不改变原来的top[stackNum]  int index = top[stackNum] - 3;  return stack[index];  }  public boolean isEmpty(int stackNum) {  //判断这个栈是否为空  if (top[stackNum] -3 < 0) {  return true;  }  return false;  }  } |