LeetCode 面试题 03.01. 三合一

作者：matthewhan
原文链接：https://ld246.com/article/1595581213336
来源网站：链滴

Problem Description

三合一。描述如何只用一个数组来实现三个栈。

你应该实现：

1. push(stackNum, value)
2. pop(stackNum)
3. isEmpty(stackNum)
4. peek(stackNum)

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。

e.g.

- 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]
```

Solution

用数组来实现栈是非常简单的，不过这里需要实现3个栈且只能用一个数组。那肯定是需要将这个数分段利用指针去管理了。

- pop方法只要操作head指针的size即可：大于0，自减；小于等于0，return -1；
- peek方法类似 pop方法；
- push方法，判断head指针的size，有空余，则移动指针去塞入新数据；
- isEmpty方法，判断head指针的size，为0则为空。

```
public class ThreeInOne {

    private final int[] data;
    private final int size;
    private final int capacity;

    /**
     * 执行用时：  13 ms , 在所有 Java 提交中击败了 56.62% 的用户
     * 内存消耗：  49.1 MB , 在所有 Java 提交中击败了 100.00% 的用户
     *
     * @param stackSize
     */
    public ThreeInOne(int stackSize) {
        capacity = stackSize * 3;
        size = capacity + 3;
        data = new int[size];
    }

    public void push(int stackNum, int value) {
        int index = headIndex(stackNum);
        System.out.println("index = " + index);

        // 头指针，容量计数功能，如果超过容量，就塞不了
        if (data[index] < capacity / 3) {
            data[index]++;
            // 剩余位置塞入数据
            data[index + data[index]] = value;
        }
```

```
}

public int pop(int stackNum) {
    int index = headIndex(stackNum);
    // stack为空
    if (data[index] == 0) {
        return -1;
    } else {
        int temp = data[index + data[index]];
        data[index]--;
        // 这里并不需要真的删除这个元素
        return temp;
    }
}

public int peek(int stackNum) {
    int index = headIndex(stackNum);
    if (data[index] == 0) {
        return -1;
    } else {
        return data[index + data[index]];
    }
}

public boolean isEmpty(int stackNum) {
    return data[headIndex(stackNum)] == 0;
}

/**
 * 根据stack下标取头指针
 * @param stackNum
 * @return
 */
public int headIndex(int stackNum) {
    int index = 0;
    switch (stackNum) {
        case 1:
            index = capacity / 3 + 1;
            break;
        case 2:
            index = size - 1 - capacity / 3;
            break;
        default:
    }
    return index;
}

@Override
public String toString() {
    return "ThreeInOne{" +
            "data=" + Arrays.toString(data) +
            ", size=" + size +
            ", capacity=" + capacity +
            '}';
```

```
    }
}
```