# 题目

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]

提示：

0 <= stackNum <= 2

# 分析

## 方法一：数组

思路：

可以使用一个数组来实现三个栈，每个栈有固定的大小。

代码：

class TripleInOne {

private:

vector<int> data;

vector<int> sizes;

int stackSize;

public:

TripleInOne(int stackSize) {

this->stackSize = stackSize;

data.resize(stackSize \* 3);

sizes = {0, 0, 0};

}

void push(int stackNum, int value) {

if (sizes[stackNum] < stackSize) {

data[stackNum \* stackSize + sizes[stackNum]] = value;

sizes[stackNum]++;

}

}

int pop(int stackNum) {

if (sizes[stackNum] == 0) {

return -1;

} else {

sizes[stackNum]--;

return data[stackNum \* stackSize + sizes[stackNum]];

}

}

int peek(int stackNum) {

if (sizes[stackNum] == 0) {

return -1;

} else {

return data[stackNum \* stackSize + sizes[stackNum] - 1];

}

}

bool isEmpty(int stackNum) {

return sizes[stackNum] == 0;

}

};

这个实现中，使用一个数组data来存储三个栈的元素，使用另一个数组sizes来记录每个栈的大小。在push方法中，首先检查栈是否已满，然后将元素压入对应的栈中，并更新栈的大小。在pop方法中，首先检查栈是否为空，然后将栈顶元素弹出并返回，同时更新栈的大小。在peek方法中，返回栈顶元素但不删除。在isEmpty方法中，判断栈是否为空。